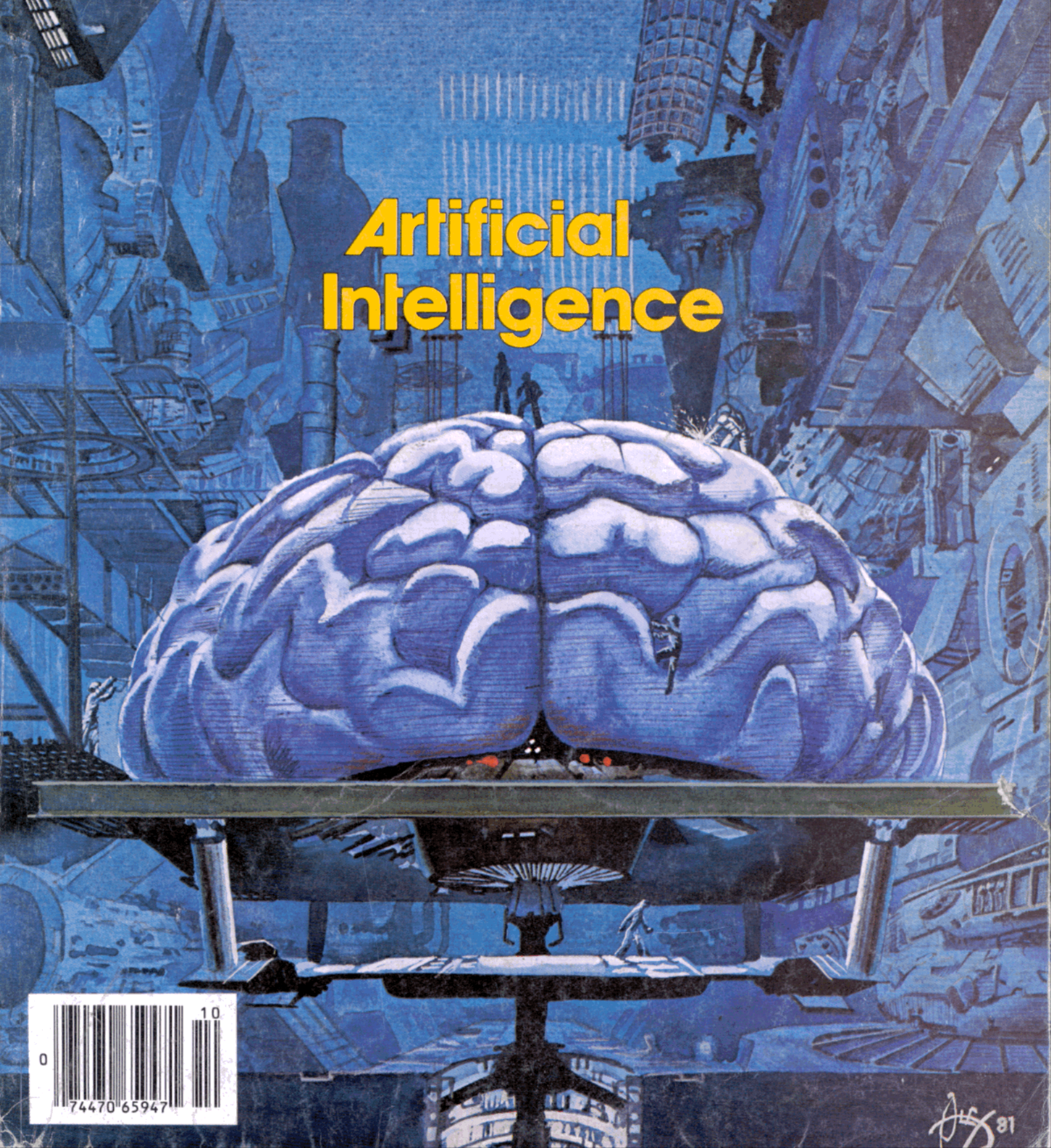
October 1981
US $2.50

80 microcomputing

the magazine for TRS-80* users

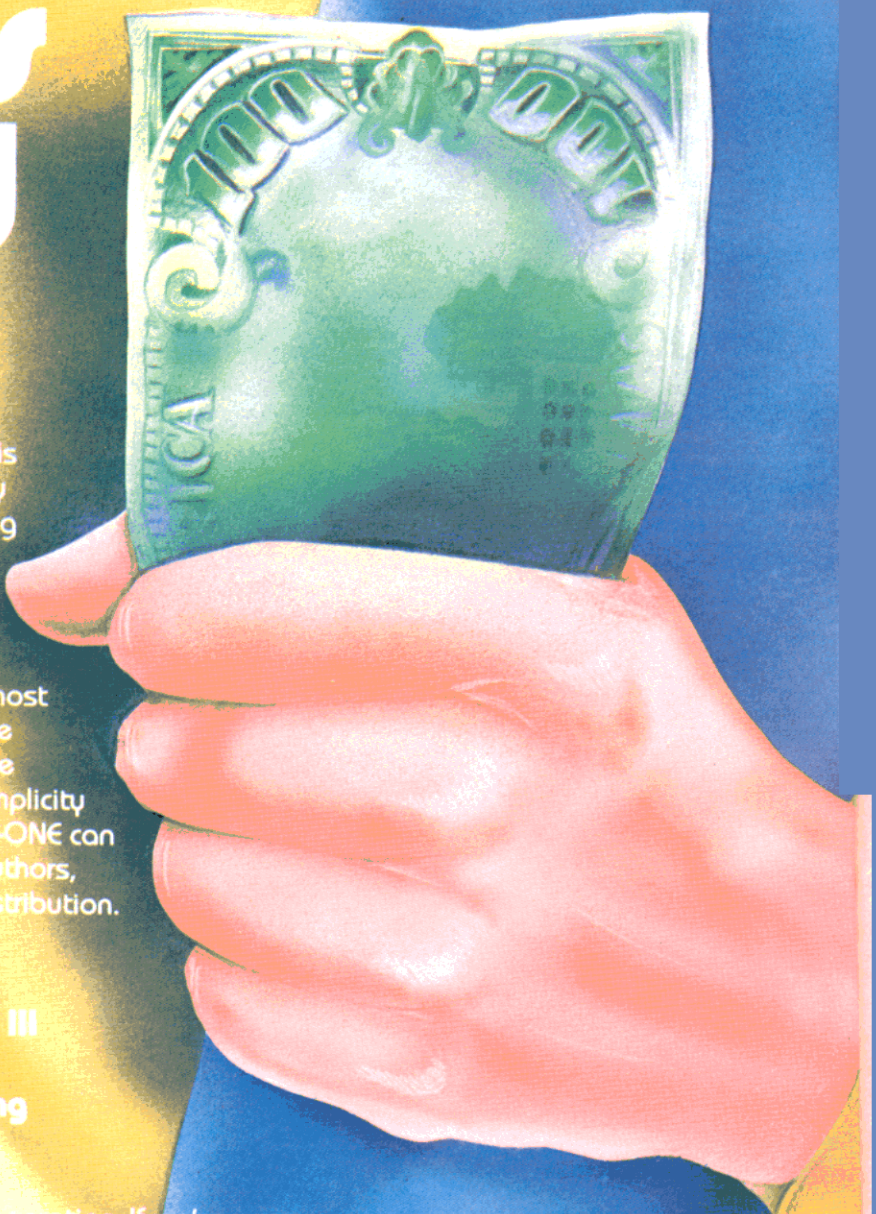
A WAYNE GREEN PUBLICATION

Artificial Intelligence



Take this hundred

...if you can



This could be yours, but probably never will. This $100 bill is the reward we offer for the discovery of a legitimate bug in a DOSPLUS DISK Operating System. To qualify you must be:

1. A registered DOSPLUS owner
2. Able to demonstrate and document a real bug, not simply a design shortcoming.

We're convinced that DOSPLUS is the fastest, most powerful, and easiest to operate system on the market. DOSPLUS works! And works right! For the businessman and hobbyists, the speed and simplicity cannot be beat. For the BASIC programmer, NO-ONE can offer you more than we do. And for software authors, compatible kernel systems are available for distribution.

NO-ONE ELSE can give you!

- **The same high level of Model I/Model III diskette and program compatibility.**
- **True, complete BASIC program chaining with files open and variables saved.**

We have it all. Features galore AND error-free operation. If we're wrong, and you can prove it, we've got a brand new $100 bill for you.

We're not conceited, we're convinced!
Order yours today, and soon you will be too!!

MICRO SYSTEMS SOFTWARE
5846 Funston Street, Hollywood, Florida 33023
Phone: (305) 983-3390

DOSPLUS

VISA master charge

CALL TOLL FREE FOR FAST SERVICE (800) 824-7888, OPERATOR 193FOR VISA/MASTERCHARGE/C.O.D. ORDERS
California dial (800) 852-7777, Operator 193, Alaska and Hawaii dial (800) 824-7919, Operator 193
TOLL FREE LINES WILL ACCEPT ORDERS ONLY!
For Applications and Technical information call (305) 983-3390 or drop us a card.

Dealers inquiries invited

TRS-80* COMPUTING EDITION

©1981 Percom Data Co., Inc. . **The Percom Peripheral** 35 cents

Percom's DOUBLER II™ tolerates wide variations in media, drives

GARLAND, TEXAS — May 22, 1981 — Harold Mauch, president of Percom Data Company, announced here today that an improved version of the Company's innovative DOUBLER™ adapter, a double-density plug-in module for TRS-80* Model I computers, is now available.

Reflecting design refinements based on both theoretical analyses and field testing, the DOUBLER II™, so named, permits even greater tolerance in variations among media and drives than the previous design.

Like the original DOUBLER, the DOUBLER II plugs into the drive controller IC socket of a TRS-80 Model I Expansion Interface and permits a user to run either single- or double-density diskettes on a Model I.

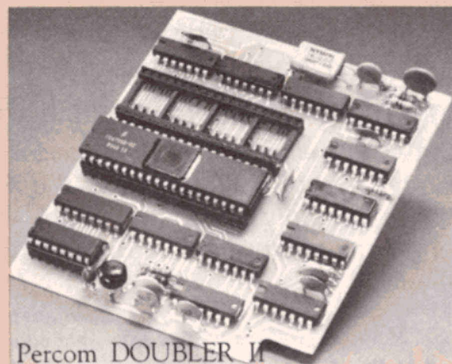
With a DOUBLER II installed, over four times more formatted data — as much as 364 Kbytes — can be stored on one side of a five-inch diskette than can be stored using a standard Tandy Model I drive system.

Moreover, a DOUBLER II equips a Model I with the hardware required to run Model III diskettes.

(*Ed. Note:* See "OS-80™: Bridging the TRS-80* software compatibility gap" elsewhere on this page.)

The critical clock-data separation circuitry of the DOUBLER II is a proprietary design called a ROM-programmed digital phase-lock loop data separator.

According to Mauch, this design is more tolerant of differences from diskette to diskette and drive to drive, and also provides immunity to performance degradation caused by circuit component aging.



Percom DOUBLER II

Mauch said "A DOUBLER II will operate just as reliably two years after it is installed as it will two days after installation."

The digital phase-lock loop also eliminates the need for trimmer adjustments typical of analog phase-lock loop circuits.

"You plug in a Percom DOUBLER II and then forget it," he said.

The DOUBLER II also features a refined Write Precompensation circuit that more effectively minimizes the phenomena of bit- and peak-shifting, a reliability-impairing characteristic of magnetic data recording.

The DOUBLER II, which is fully software compatible with the previous DOUBLER, is supplied with DBLDOS™, a TRSDOS*-compatible disk operating system.

The DOUBLER II sells for $2~~[illegible]~~5, including the DBLDOS diskette. NOW $169.95!

1

Owners of original DOUBLERs may purchase a DOUBLER II upgrade kit, without the disk controller IC, for $30.00. Proof of purchase of an original DOUBLER is required, and each DOUBLER owner may purchase only one DOUBLER II at the $30.00 price.

The Percom DOUBLER II is available from authorized Percom retailers, or may be ordered direct from the factory. The factory toll-free order number is 1-800-527-1592.

Ed. note: Opening the TRS-80 Expansion Interface may void the Tandy limited 90-day warranty.

Circle 258 on inquiry card.

Circuit misapplication causes diskette read, format problems.

High resolution key to reliable data separation

GARLAND, TEXAS — The Percom SEPARATOR™ does very well for the Radio Shack TRS-80* Model I computer what the Tandy disk controller does poorly at best: *reliably* separates clock and data signals during disk-read operations.

Unreliable data-clock separation causes format verification failures and repeated read retries.

CRC ERROR–TRACK LOCKED OUT

The problem is most severe on high-number (high-density) inner file tracks.

As reported earlier, the clock-data separation problem was traced by Percom to misapplication of the internal separator of the 1771 drive controller IC used in the Model I.

The Percom Separator substitutes a *high-resolution* digital data separator circuit, one which operates at 16 megahertz, for the low-resolution one-megahertz circuit of the Tandy design.

Separator circuits that operate at lower frequencies — for example, two- or four-megahertz — were found by Percom to provide only marginally improved performance over the original Tandy circuit.

The Percom solution is a simple adapter that plugs into the drive controller of the Expansion Interface (EI).

Not a kit — some vendors supply an untested separator kit of resistors, ICs and other paraphernalia that may be installed by modifying the computer — the Percom SEPARATOR is a fully assembled, fully tested plug-in module.

Installation involves merely plugging the SEPARATOR into the Model I EI disk controller chip socket, and plugging the controller chip into a socket on the SEPARATOR.

The SEPARATOR, which sells for only $29.95, may be purchased from authorized Percom retailers or ordered directly from the factory. The factory toll-free order number is 1-800-527-1592.

Ed. note: Opening the TRS-80 Expansion Interface may void the Tandy limited 90-day warranty.

Circle 508 on inquiry card.

All that glitters is not gold

OS-80™ Bridging the TRS-80* software compatibility gap

Compatibility between TRS-80* Model I diskettes and the new Model III is about as genuine as a gold-plated lead Krugerrand.

True, Model I TRSDOS* diskettes can be *read* on a Model III. But first they must be converted and re-recorded for Model III operation.

And you *cannot write* to a Model I TRSDOS* diskette. Not with a Model III. You cannot add a file. Delete a file. Or in any way modify a Model I TRSDOS diskette with a Model III computer.

Furthermore, your converted TRSDOS diskettes cannot be converted *back* for Model I operation.

TRSDOS is a one-way street. And there's no retreating. A point to consider before switching the company's payroll to your new Model III.

Real software compatibility should allow the *direct, immediate* interchangeability of Model I and Model III diskettes. No read-only limitations, no conversion/re-recording steps and no chance to be left high and dry with Model III diskettes that can't be run on a Model I.

What's the answer? The answer is Percom's OS-80™ family of TRS-80 disk operating systems.

OS-80 programs allow *direct, immediate* interchangeability of Model I and Model III diskettes.

You can run Model I single-density diskettes on a Model III; install Percom's plug-in DOUBLER™ adapter in your Model I, and you can run double-density Model III diskettes on a Model I.

There's no conversion, no re-recording.

Slip an OS-80 diskette out of your Model I and insert it directly in a Model III.

And vice-versa.

Just have the correct OS-80 disk operating system — OS-80, OS-80D or OS-80/III — in each computer.

Moreover, with OS-80 systems, you can add, delete, and update files. You can read *and write* diskettes regardless of the system of origin.

OS-80 is the original Percom TRS-80 DOS for BASIC programmers.

Even OS-80 utilities are written in BASIC.

OS-80 is the Percom system about which a user wrote, in Creative Computing magazine, ". . . the best $30.00 you will ever spend."†

Requiring only seven Kbytes of memory, OS-80 disk operating systems reside completely in RAM. There's no need to dedicate a drive exclusively for a system diskette.

And, unlike TRSDOS, you *can* work at the track sector level, defining and controlling data formats — in BASIC — to create simple or complex data structures that execute more quickly than TRSDOS files.

The Percom OS-80 DOS supports single-density operation of the Model I computer — price is $29.95; the OS-80D supports double-density operation of Model I computers equipped with a DOUBLER or DOUBLER II; and, OS-80/III — for the Model III of course — supports both single- and double-density operation. OS-80D and OS-80/III each sell for $49.95.

429

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE. PRICES DO NOT INCLUDE HANDLING AND SHIPPING.

PERCOM DATA COMPANY, INC. 211 N. Kirby Street Garland, Texas 75042 (214) 272-3421

80 Contents

PUBLISHER
Wayne Green

EXECUTIVE VICE PRESIDENT
Sherry Smythe

ASSISTANT TO PRESIDENT
Matt Smith

CORPORATE CONTROLLER
Roy Johnson

ASSOCIATE PUBLISHER
Edward Ferman

ASSISTANT PUBLISHER
Jeff DeTray

DIRECTOR OF MARKETING
(603) 924-7296
Debra Boudrieau

BULK SALES MANAGER
Ginny Boudrieau

ADVERTISING SALES
(603) 924-7138
John Gancarz
Hal Stephens

COORDINATION
Penny Brooks

OFFICE MANAGER
Marcia Stone

Paid Audited Circulation

80 Microcomputing (ISSN -0199-6789) is published monthly by 1001001 Inc., 80 Pine St., Peterborough NH 03458. Phone: 603-924-3873. Second class postage paid at Peterborough, NH, and additional mailing offices. Subscription rates in U.S. are $18 for one year and $45 for three years. In Canada, $20 – one year only, U.S. funds. Canadian distributor: Micro Distributing, 409 Queen St. West, Toronto, Ontario, Canada M5V 2A5. BC Canadian distributor: Graymar Data Services, Ltd., #4 258 E. 1st Ave., Vancouver, BC V5T 1A6. Foreign subscriptions (surface mail), $28 – one year only, U.S. funds. Foreign subscriptions (air mail), $60 – one year only, U.S. funds. In Europe contact Monika Nedela, Markstr. 3, D-7778 Markdorf, W. Germany. In South Africa contact *80 Microcomputing*, P.O. Box 782815, Sandton, South Africa 2146. All U.S. subscription correspondence should be addressed to *80 Microcomputing*, Subscription Department, P.O. Box 981, Farmingdale, NY 11737. Please include your address label with any correspondence. Postmaster: Send form -3579 to *80 Microcomputing*, Subscription Services, P.O. Box 981, Farmingdale, NY 11737.

PUBLISHER/EDITOR
Wayne Green

MANAGING EDITOR
Debra Marshall

SENIOR EDITOR
Pamela Petrakos

CHIEF TECHNICAL EDITOR
Jake Commander

PRODUCTION EDITOR
Susan Gross

NEWS EDITOR
Bert Latamore

REVIEW EDITOR
Michael Nadeau

NEW PRODUCTS EDITOR
Janet Fiderio

TECHNICAL EDITORS
Chris Brown
G. Michael Vose

EDITORS
John P. Mello Jr.
Lynn Rognsvoog
Steven Frann

LAYOUT EDITORS
Joan Ahern
Bob Dukette
Sharon Phinney
Sue Symonds
Anne Vadeboncoeur

TECHNICAL CONTRIBUTING EDITOR
Dennis Kitsz

EDITORIAL ADMINISTRATION
Pat Graham
Nancy Noyd

DESIGN ASSOCIATE
Denzel Dyer

PRODUCTION MANAGER
Nancy Salmon

ASST. PRODUCTION MANAGER
Michael Murphy

AD GRAPHICS MANAGER
Robert Drew

AD COORDINATOR
Patty Mackowsky

ADVERTISING PRODUCTION
Steve Baldwin, Bruce Hedin, Jan Preston, Dennis Christensen

PRODUCTION DEPT.
Fiona Davies, Linda Drew, Sandra Dukette, Kenneth Jackson, Theresa Ostebo, Dianne Ritson, Deborah Stone, Judy Wimberly, Donna Wohlfarth

PHOTOGRAPHY
William Heydolph, Terrie Anderson, Bill Suttenfield, Paul Babich, Thomas Villeneuve

TYPESETTING
Sara Bedell, Michele Desrocher, David Hayward, Stephen Jewett, Mary Kinzel, Ann Lord, Kelly Smith, Karen Stewart

Cover by Alex Stevens

TRS-80*
Model I
Model III
Products that set Precedents.
OMIKRON™
COUGAR
Word Processing
MAPPER CP/M Adaptor
CBASIC II*** CP/M** AND
THE BEST WORD PROCESSOR
$1000.00 VALUE ONLY $ 299
PHOENIX
UNPRECEDENTED SAVINGS
are available on selected CP/M programs through COUGAR, Omikron's official users group.
TRADEMARKS:
*TRS-80/Radio Shack/Tandy Corp.
**CP M/Digital Research
***CBASIC II/Compiler Systems
****PHOENIX/Palantir Inc.
OMIKRON
1127 Hearst St.
Berkeley, CA 94702
(415) 845-8013
367

80 REMARKS

by Wayne Green

"Before . . . computer artificial intelligence . . . we have to come to terms with what we mean by 'intelligence.'"

Artificial Intelligence, Eh?

That gets the same sort of snickering laugh from me that I provide the robot-oriented folk. You know how to make a robot, don't you? Well, you start with a British midget . . .

Before we can tackle computer artificial intelligence we have to come to some terms with what we mean by "intelligence." Frankly, even with our largest computers we are still far away from anything remotely describable as intelligence. As measured by psychologists, intelligence has to do with the ability to cope with a new situation: This is the opposite of the normal computer approach. A computer can only cope with those things for which it has been designed to cope.

In addition to our brains being several orders of magnitude more complex than even our best computers, there may be other more serious obstacles in the way of creative problem solving by computers. Take the simple matter of memory—have you any idea how your brain stores information? I hope you said no, because no one yet has even a hint as to how, or where, we store all the data we pick up.

If you add up the amount of input to the brain—from the zillions of rods and cones in the eye, at so many frames per second; from the ears, nose, touch all over the body; temperature; kinesthesia; pressure; and so on—the amount of material put into storage is incredible. We seem to forget most of this input, but under hypnosis we are able to regress and get back full perception from any time. The information is recorded; it is only our ability to contact the recordings which is a problem.

I remember when a chap first came out with the announcement that it was possible to go back in memory to the prenatal period. Boy, did he get the hee-haw. Today it is commonly accepted, complete with the ability of anyone not only to recall events but to recall voices and things that were said near one during that period of life. I've personally regressed many people to the prenatal period and had them play back their recorded memories. Often these memories have a surprising impact on present day life.

Prenatal memories are unlike later ones, in that they are mere recordings of perceptions and can only be recalled under hypnosis, bypassing the conscious mind entirely. The prenatal conciousness didn't know what the sounds meant, so they were not interpreted and filed under as complex a cross index system as are later memories.

Getting back to memory, it appears we are able to store considerably more data than the brain can possibly hold. When we touch some parts of the brain certain memories are stimulated, but this may be because we have hit a switching circuit rather than a memory storage area. Removing large hunks of the brain does not always remove memory.

It may be that we are able to use a holographic type of storage system, somehow storing large numbers of memories or a few molecules. But I'm skeptical about that; in view of our ability to recall things throughout our lifetime with amazing clarity, in spite of our cells being constantly replaced, I wonder if we aren't using some outside storage medium.

If you've read much about dying and death, you are aware that we have an incredible body of evidence that there is part of our being called the spiritual which seems to continue after death. This part of us seems to retain all those memories which were supposedly buried with the body. So, there might be some sort of limitless, timeless storage system we use. Physical problems with the brain—or programming problems—can interfere with our memory access. Hypnosis and other sneaky techniques known to psychologists make it relatively simple to access even the most hidden (by programming) memories.

If I'm right about this, our electronic computers are going to have a heck of a job matching us in memory capacity, even with laser disks. And then comes the complex system of indexing we use. Perhaps you have to have done psychological work to appreciate how perfect an instrument the brain is and how invariably it will respond to a request on the unconscious level. If I get you to relax and ask you to return to a time when you were eating a ham sandwich, your mind will—infallibly—do just that. If I ask you to return to an earlier ham sandwich, it will. And I can take you back to the first ham sandwich you ever ate. It's all there, when you know how to access it. All cross-indexed beyond the wildest hopes of any computer imagined today.

That brings us back to the concept of intelligence—the ability to cope with a new situation on the basis of past information. The built-in indexing system we have can break down a new situation into extremely small parts, all on an unconscious level, and develop a scheme for meeting a completely new event. The better able the brain is to do this integration, the higher the intelligence. People with a high IQ have a better working computer system.

As one of the founders of Mensa, the high IQ society, I've met and talked with hundreds upon hundreds of people with high IQs. Yes, they have good computers, but that doesn't mean they appear very smart. Let's draw a parallel with one of our microcomputers. If we give our computer a very limited amount of data with which to work, and then we make sure that part of that data is not true, and then, just to really screw things up, we program the system with a crummy program, we can see that the computer itself is of less significance than we thought.

People are programmed, beginning with that prenatal time. If they are taught values which are not true, given data which is not factual, it really doesn't make any difference how fantastic their computer is, they will still do dumb things and be failures. I hope I will not make any additional enemies, but the truth is that the Mensa people are no more successful than anyone else. I have found no correlation between intelligence and success.

The Mensa people do seem to have a better computing system, but then, no one ever claimed that everyone was equal in any way. Some make a big deal out of being intelligent. But then, some people use their height in similar ways—or anything else they inherited which is deemed superior in our society.

I suppose that if we could build a computer system big enough to store as much as we can with our minds which had the ability to cross-index as well as we do, we

might then start talking in terms of artificial intelligence. Until then, please don't bug me.

If you want to take issue over the concepts of a spirit body, please do me the kindness to read the literature before discussing it. The same goes for reincarnation, communications with the dead, and all that occult stuff. And don't forget to come to grips with parapsychology, mental communications, ghosts and other psychic phenomena.■

Regaining Our Technology

It should be no secret that Japan has passed us by in much of our electronic technology. I've written a good deal about that in the last few months, and the reason I think it became possible. Part has to do with the support the Japanese government gives their electronic industries, the tax benefits, the low cost loans, and the skilled training for their workers. But part of the problem, I'm convinced, lies in a little-known event that occurred almost twenty years ago.

In 1963 proposals were made to change the ham rules. With about 85 percent of the hams believing they would have to take the difficult FCC exams over to get back their frequencies, spirits dropped and amateur radio growth stopped dead—for over ten years!

If ham growth had continued as it had from the end of WWII, we would today have over two million hams and over half of these would be engineers and technicians, working in our electronics and communications firms. Well, we lost out on about one million engineers and technicians and there is no way that the loss of that many technical people—particularly hams, who are the most rabid of the lot—hasn't hurt our technology.

At the same time we stopped our ham growth, the Japanese set up a no Morse Code license and their ham population has far surpassed ours, even though they have only half our population. When I visit the computer labs in Japan I am greeted as W2NSD and known through *73 Magazine*. I see hams all through their labs.

Remember, if you will, that virtually every major communications breakthrough has been discovered and pioneered by hams. FM was pioneered by hams, NBFM was too, as was sideband, RTTY, and so on.

The new generation of computer hackers may, to some degree, help us through this technician famine, but we need to do something on a national level to get interest in electronics, computers and so on into our high schools and infect our 14 and 15 year olds with the ham and computer virus. We are way behind now, so it is going to take a lot of work to catch up. I spoke recently to the chairman of the FCC about this and there is a good possibility we may be able to get this to the White House for consideration.

One college can't turn out a million technicians, but if we can get amateur radio going again—perhaps with a version of a no-code license—and we can get schools to push computers, we could catch up to the Japanese by 1990. We're not going to do it by accident, that's for sure.■

Micro Mountain

About a year ago I got to talking with the presidents of two local colleges about my plans for introducing microcomputer courses into their educational curriculum. Both liked the idea and we started to lay the groundwork.

Then I got a call from one of them saying he wanted to get out of the college business and wondered if Wayne Green Inc. was in any position to take on a college. We weren't at that time, but I started thinking seriously about it and looking for someone with the background to tackle the job. You don't find someone with the ability to run a college quickly, so the college was eventually sold to a Florida school.

The college was a nice one, with 800 acres and 15 major buildings, but it had drawbacks, too. It had been built in the days of cheap energy and was a bear to keep heated in the winter. The spread-out buildings meant extensive plowing after every snow storm—and so on.

Just recently we lucked into a chap with the right background to get our school started. Since then I've been giving a lot of thought to the plans and they have been growing almost daily. A recent NBC White Paper on training workers fueled the fires of imagination.

Since 80 percent of all new jobs come from small firms—particularly new firms—why not think in terms of an industrial area centered around the school? This would allow entrepreneurs to open small firms and make use of students as part of the teaching process, providing workers of outstanding intelligence and skills at a low cost.

I had been thinking in terms of a school which would have an 18-24 month course, resulting in an Associate's Degree. Part of that time would be spent learning theory and the other part doing practical professional work, either with Instant Software or with one of the other nearby commercial firms. Thus, each student would get the benefit of practical on-the-job training as well as theory and associated business courses.

Could we build an eastern counterpart to Silicon Valley, calling it Micro Mountain? After talking with the people at Atari and Apple and smaller firms in the microcomputer business, there is no question in my mind about the need for skilled people: The industry needs them badly. Indeed, the lack of skilled workers has been seriously holding back growth of the field and the situation is only going to get worse.

The courses I had in mind for the school would start with an understanding of microcomputers: how they work; how to fix them; how to interconnect accessories and get them to work. We'd explain the architecture of the popular chips and the reasons for support chips. I'll bet we could get some help with these courses from chip manufacturers—they need skilled people too.

Then we would go on to printers, disk drives, and other accessories: how they work, how to fix them, and why they have been designed the way they were.

Software would come next, with introductions to all common languages and heavy emphasis on Basic and perhaps Pascal. We'd go into operating systems, machine and Assembly programming, utilities and how they simplify programming and service of systems. Then we would get into teaching about the many applications programs—how they work, what to look for, how to fix them. We'd work with as many different word processors as possible, accounting packages, and industry-specific packages.

Man does not live by computers alone, so we would also teach a variety of business programs such as business law, how to buy or rent buildings, personnel management, how to apply for and get a job, finance, accounting and bookkeeping, advertising, promotion, writing, editing, printing, marketing, packaging, Z-theory, photography, and so on. We want students to be able to go to work for a firm and be qualified to proceed to upper management.

The idea seems like one whose time has come, so we'll be developing it and looking for support from the industry.■

Computers and Hamming

With almost 30 percent of the active hams already into computers it is a wonder so little has been done to marry the two interests. One of my other publications, *73 Magazine* (for radio amateurs), is going to take the big step and put the two together with an on-the-air ham bulletin board system.

The idea is to have one channel where

Continues on page 67

80 INPUT

"Those of us who have been using Model IIIs. . .know what frustration is all about."

READY or Not. . .

I am writing in response to Ron Balewski's article "Never Ready" (July 1981). He stated in the article that the message string that was to be displayed instead of READY would not appear after CLOAD because it uses a different address for re-entry to Basic.

But, it is possible to make this patch into the RAM location that is called upon re-entry to Basic. So, I have written a very short routine to handle the flaw in the program. It only uses 5 bytes, thanks to a little ASCII character called the upward line-feed (27 decimal or 1B hex). I have never seen this character used in a program before although it seems to be a very handy character to use. This is the routine:

```
LD      A,1BH  ;Up-ward Line-feed
CALL 33H       ;Display at cursor
```

This requires the following changes in the Basic program:

```
30 FOR K = 32683 TO 32703:READ D:POKE K,D:NEXT K
```

And add the following lines:

```
35 DATA 62,27,205,51,0
125  K = 16821:POKE  K,  195:POKE  K + 1,171:POKE
K + ,127
```

Be sure to set the memory size to 32683 instead of 32688 before running the program.

Craig Riecke
Lincoln, NE

What About Level I

I have just read the June 1981 issue of *80 Microcomputing*, the first subscription copy that I have received (the transatlantic postal service is not as efficient as it could be), and I must say that I was very impressed. It is fair to say that a single issue contains more information on the TRS-80 than the three leading UK microcomputer publications combined in an entire year.

But. . . there wasn't a single item on or program in Level I Basic. Surely some of your readers have TRS-80s equipped with Level I? I know the general idea is that Level I is OK for starting with and Level II or Model III Basic is required for serious use, but it depends on what you mean by serious. My Level I system plays Chess, Star Trek, Adventure, runs the Editor/Assembler and will shortly be doing some simple word processing, thanks to the Electric Pencil. In fact, it does everything I want a personal computer to do, so why bother upgrading to Level II? (I should add that I am fully familiar with the facilities of Level II and Disk Basic).

About a year ago I helped form the UK Level I User Group, which is dedicated to TRS-80s equipped with Level I Basic, and to making them do more than people think they are capable of. The group's main activity is the production of a bi-monthly newsletter/magazine, and fellow *80 Microcomputing* readers are invited to write to me for a sample copy (free, but please enclose $1 for postage).

Nick Rushton
123 Roughwood Drive
Northwood, Kirkby
Merseyside L33 9UG
United Kingdom

Pilot Praise

The following is an open letter to Randy Hawkins, author of "Pilot—The Language of Computer Aided Instruction" (July 1981).

I've dealt with all kinds of people over the years, but never anyone who was as prompt and courteous as you. When you told me last Saturday you'd mail a fix for your Pilot program on Monday, I expected that would be the last I'd hear from you. . . at best it would be weeks. But this afternoon's mail brought your four-page Model III changes.

Next, don't apologize for inconveniences. Those of us who have been using Model IIIs for some time know what frustration is all about.

Now, for some items on how it's working for me. After debugging it, I found the system tape wouldn't load. I sat here staring at the ** on the screen and watching it switch to D* at the very end of the program. I made extra copies. Normally, the D* message tells me I've got a problem with volume settings, although it's mighty rare on the Model III. If it persists, I clean the heads, then normally it boots up without any trouble.

But your Pilot 80 sat there sticking its D* tongue out at me every time, no matter how I adjusted the volume. Finally, a little bell went "tinkle-tinkle" in my head. This month's *Microcomputer News* (from Tandy), mentioned that Haunted House will load in the Model III if you simply load it over itself on a checksum error. Next time around, I watched the ** change to D*, then a pause while it passed the blank portion between saves, then another series of blinking D*s, and danged if it didn't perform like a charmer!

I found when putting in the tutorial program that your suggestion to use "a" instead of "@" or the shifted "@" was of little help. It worked fine everywhere except in line 325. I went over that damned line at least 20 times; I edited it, rewrote it, deleted it and rewrote it, and no matter how hard I tried, I couldn't get that "a" to appear on the screen. It came out "A" every time. Well, you warned us that using two quotations in a line could lead to unpredictable results. Have you ever edited a line, hit L several times and saw it was letter perfect, then entered and listed it? Comes up "A".

My final attempt was to delete lines 305 to 330, name the program to the end, and then redo the missing lines, this time combining lines 320 and 325. Now I had *three* quotes in one line. But it worked perfectly.

I can assure you Pilot will make some of my self-imposed tasks easier, and produce neater programs when I want to create a tutorial. Even though my system is 48K and I can't make use of 16K of my RAM, I take my hat off to you for an excellent program. I have also stuck your name in the master tape where it belongs to identify your work.

Karl H. Meyer
Corpus Christi, TX

Bungled Broker

Your article "The Software Broker" (June 1981) contained many serious mistakes. The following are some that I have found. There may be more.

Continued on page 14

80 AID

Break Solution

In reference to TRSDOS POKE (June 1981), defeat the Break key under TRSDOS 2.1/2.2/2.3, NEWDOS 21 or ULTRADOS, I use the following line: POKE PEEK (17171) + PEEK (17172) *256 + 1, 0. To turn on: POKE PEEK (17171) + PEEK (17172) *256 + 1, 1

Please note this does *not* work with NEWDOS 80, DOSPLUS nor ULTRA-II. These new systems have the ability to defeat the Break key via a system or library command.

Vernon B. Hester
42403 Old Bridge Road
Canton, MI 48188

Data Inputs Wanted

Would it be possible to complement a wonderful program ("The Software Broker," June 1981, p. 268) with a short explanation of how to prepare the data disks (mainly the stock price data and the index values)?

I don't understand how you can do it from lines 300-380 on Program Listing 1. The variable D9A will accept only numerical data—what happens to the name of the stock, for example.

In spite of the tremendous amount of space that you dedicated to this program I think it would have been very nice to have a sample of data inputs at the end of the program.

Harold E. Lange
P. O. Box 450182
Miami, FL 33145

Disappearing Act?

About a year ago I purchased a machine language program on cassette for my TRS-80, which was similar to the arcade or "Space War" game for Apple. Since that time I have searched for another copy as the tape worked twice, then would not load. In trying to track down the original supplier (I think the name was "creative software" or "creative games" or something similar) there was no phone number and returning the tape for replacement only resulted in a reply of "addressee unknown." I therefore assumed the company had gone out of business.

If anyone can supply information concerning the demise of this company, or of a new supplier, it would be greatly appreciated.

Douglas C. McMillan
105 Burlington Beach
Valparaiso, IN 46383

OM Error Fix

I'm writing in regard to my letter published in July 1981 concerning the OM error I received on loading KBEEPFIX. The response was outstanding; I would like to thank all those who wrote to me. It appears the answer is to JP to O6CCH, rather than to 1A19. At least this gets rid of the dreaded OM error.

Now I have a new question. Is there anyone who knows of an add-on disk controller unit other than the expansion interface. (I am mainly interested in an already assembled, ready to plug in unit—not one that has to be assembled or searching for parts as in the LNW board). I would like to hear from any one who knows of a unit.

Bernard F. Gaffney Jr.
524 Riley St.
Lansing, MI 48910

POKEing Along

In reference to your column, *80 Aid*, (June 1981), about the various POKEs required to disable the Break key, I have done a bit (pardon the pun) of research and created a list of POKEs for Level II and various DOSers, including TRSDOS versions 2.1 through 2.3, Apparat's NEWDOS PLUS, and Apparat's NEWDOS 80.

Robert Churchill
2390 California St.
Saginaw, MI 48601

IRV Solution

This is in response to the letter of Rev. Richard W. Beebe (March 1981) regarding problems getting IRV to work with TRSDOS 2.3. I had similar problems with NEWDOS but found a way out. I tried this solution with TRSDOS 2.1 and it works. (Sorry, but with all the well-known problems of 2.1, I never bothered to upgrade to 2.3, settling instead for NEWDOS.) I figure if anything, it will work with his TRSDOS 2.3.

His best bet is to get IRV into RAM using TRSDOS command Load IRV, then invoking Debug and pressing the Break key to activate Debug; then get into IRV by entering the Debug command GAAAA. AAAA equals the execute process on page 5 of the IRV brochure (e.g., for his 48K system it would be GFD36).

Incidently, when IRV is operational it does not work with Electric Pencil and will reboot the system when returning to DOS. It also disables the chain command of NEWDOS 80. If anybody has figured patches for these problems I would welcome them very much. IRV is a very powerful utility and I bet it has still many undiscovered uses.

And speaking of Electric Pencil and the NEWDOS-80 chain command, a DOS command file can be made by Pencil, saved on disk as filename/PCL then activated by chain filename/PCL.

Alan H. Hyde Jaimovich, MD
440 West Foothill Blvd.
Glendora, CA 91740

Printing a Disk Directory

I am looking for information on how to get a disk directory printed on a printer. I have a TRS-80 Model I, and I am using TRSDOS 2.3. It would be very helpful in organizing my disks. I would appreciate any information.

Michiel van de Panne
Box 13, Site 16, SS1
Calgary, Alberta T2M 4N3
Canada

META TECHNOLOGIES

26111 Brush Avenue, Euclid Ohio 44132
CALL TOLL FREE 1-800-321-3552 TO ORDER
IN OHIO, call (216) 289-7500 (COLLECT)

MTC AIDS-III™

MODEL I . . . $69.95 **MODEL II . . . $99.95**

Introducing the latest addition to MTC's family of data management systems, AIDS-III. NO PROGRAMMING, easy to use. COMPLETE PACKAGE including demonstration application, documentation and MAPS-III (see below).

- Up to 20 USER-DEFINED FIELDS of either numeric- or character-type.
- CHARACTER-type fields may be any length (total: up to 254 characters).
- NUMERIC-type fields feature automatic formatting, rounding, decimal alignment and validation.
- Full feature EDITING when adding or changing records:
  - ENTER FIELD (can't type in more characters than specified).
  - BACKSPACE (delete last character typed).
  - DELETE FIELD contents.
  - RESTORE FIELD contents.
  - RIGHT-JUSTIFY FIELD contents.
  - SKIP FIELD (to next or previous field).
  - SKIP RECORD (to next or previous record).
- SORTING of records is MACHINE CODE assisted.
  - 200 RECORDS (40 characters) in about 5 SECONDS.
  - ANY COMBINATION of fields (including numerics) with each field in ascending or descending order.
- SELECTION of records for Loading, Updating, Deleting, Printing and Saving is MACHINE CODE assisted.
  - Specify up to 4 CRITERIA, each using one of 6 RELATIONAL COMPARISONS.
  - LOAD or SAVE selected records using MULTIPLE FILES.
  - Example: "Select records representing those people who live in the state of Colorado, but not in the city of Denver, whose last names begin with "F" and whose incomes exceed $9000.00.
  - Example: Select records representing those sales made to XYZ COMPANY that exceed $25.00, between the dates 03/15 and 04/10.

MAPS-III (MTC AIDS PRINT SUBSYSTEM), included at no charge, has the following features:

- Full AIDS-III SELECTION capabilities.
- Prints user-specified fields DOWN THE PAGE.
- Prints user-specified fields in titled, columnar REPORT FORMAT, automatically generating column headings, paging and (optionally) indentation.
- Can create a single report from MULTIPLE FILES.
- Prints user-defined formats for CUSTOM LABELS, custom forms, etc.

BELOW ARE TESTIMONIALS from owners of AIDS systems. These are absolutely authentic statements and are typical of the comments we receive.

"This program will do more for my business than all the other programs I have, combined."
David Wareham, Vice President (EDP), National Hospital and Health Care Services Inc.

"We have 32 different Data Base Management packages for the TRS-80. AIDS-III is easily the best. It also makes it easier for us to step up to our Model II since the package is available for both computers."
Jack Bilinski, President, 80 Microcomputer Services

"Your AIDS program is far and away the finest information management system that I've ever seen. I am currently using it to maintain a clear picture of the demographic data on all the kids in our residential treatment program and it is working for me superbly."
Frank Boehm, Director, Front Door Residential Treatment Program

MTC AIDS CALCULATION SUBSYSTEM-III™

MODEL I . . . $24.95 **MODEL II . . . $39.95**

MTC's most popular AIDS subsystem. Use for report generation involving basic manipulation of numeric data. Features are:

- User-specified page title
- Columnar Headings
- Optional Indentation
- Use for accounting, inventory, financial and other numeric-based information systems.
- Columnar subtotals generated when there is a change in a user-specified column.
- User-specified Columnar Totals
- Columnar values computed using constants and/or column values
- Balance forward calculations (Ex: Gross sales equals previous gross sales + sale amount + sales tax).

Compare AIDS-III™/CALCS-III™ with any other data management package under $100!

CALCS-III™ REQUIRES THE PURCHASE OF AIDS-III™

AIDS OWNERS! WE HAVE WHAT YOU'VE BEEN WAITING IV. . .

MTC CALCS-IV™, that is.

- More Computations
- Save Report Formats on Disk
- Faster, and more!

MTC CALCS-IV™........... $39.95
For Model II................ $59.95

MTC AIDS MERGE-III™

This subsystem will combine up to 14 AIDS-created data files into a single, large file. An optional purge capability removes duplicate entries while performing the merge operation (can even be used to eliminate duplicates in a single file). Machine-code assisted for high-speed performance, MERGE-III™ properly handles files sorted by any combination of fields, including numerics, with each field in ascending or descending order.

MTC AIDS MERGE-III™........... $19.95
For Model II $29.95

THE COMPLETE MTC AIDS-III™ PACKAGE

SAVE $$$$

Includes MTC AIDS-III™ CALCS-III™ and MERGE-III™

A comprehensive system at a competitive price!

MODEL I & III $99.95
MODEL II $149.95

Add $25 for CALCS-IV™

MORE PRODUCTS

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation. DATALIFE is a trademark of VERBATIM. PLAIN JANE, AIDS-I, AIDS-III, CALCS-III, CALCS-IV, MERGE-III are trademarks of MTC. 1981 by Metatechnologies Corporation, Inc.

MOST ORDERS SHIPPED WITHIN ONE BUSINESS DAY

Products damaged in transit will be exchanged.

PRICES IN EFFECT October 1, 1981 THRU October 31, 1981.

Prices, Specifications, and Offerings subject to change without notice.

8110

WE ACCEPT
- VISA
- MASTER CHARGE
- CHECKS
- MONEY ORDERS
- C.O.D.

- Add $3.00 for shipping & handling
- $3.00 EXTRA for C.O.D.
- Ohio residents add 5½% sales tax.

One glaring incidence is in "UPDATE/DTA", line 1420 which reads: For P39 = 1 TO 80.

This error occurs several times. I am sure that something else was meant, but what?

Another is in "STOCK/ANA," lines 170 to 180: 170 SP = ((P(A) – PZ)[2 + SP. It should probably be: SP = ((P(A) – PZ)[2 + SP). Ditto for line 190.

In the program named "BREADTH/MKT" there is a mistake in the establishment of the value of L. I think that the problem is in lines 360–380, but not knowing what the author meant I'm not sure.

Also trading volume as written produces nothing but a series of #'s and + 's on the printer.

I am sure that there are other mistakes, but I have to get past these first. I sure do wish that these programs were tested before being published. It sure would save a lot of aggravation.

Also in the program "Real Rule of 78s" the Stop statement should not be in the program if it is to continue.

Richard Eidmann
Philadelphia, PA

Bad Listings

With the renumbering utilities around, why can't you clean up the listings you publish? S. Hunter's "Sans Disks" (April 1981, p. 186) is a case in point.

Robert McDaniel
Clifton, NJ

Your point is well taken. However, renumbering programs may involve re-writing a whole manuscript so that references to the program still make sense. It's really up to program authors to submit their programs to the magazine in as tidy a form as possible.—the Editors.

Computers for the Needy

Santuary, Inc. is a non-profit, non-sectarian organization which aids youngsters who have special problems. They are placed in the home for guidance and counseling. The home is staffed by professionals dedicated to assisting young adults. The school is a public institution which is operated on what little federal funds are available and on private donations. Needless to say, the funds are not sufficient to sustain the children's needs.

I am not affiliated with the school; however, I know its director, Sister Albina Guillory, M.S.C. She has told me of their lack of educations equipment, personnel and operating funds. She has worked actively just to keep the home open. Its over-extended facilities are in need of annexation. She says that the demands on the school far exceed its capabilities.

I suggested to her that I would write a number of microcomputing magazines for assistance from its readers. You already know the capabilities of computers in education. I know that you can see the added advantages of CAI to special students who need that added patience and encouragement that few teachers are able to devote to their students on an individual basis.

If your readers would be able to help these youngsters in any way, I know that Sister Albina would be more than grateful to them. Please remind the readers that their donations are tax deductible and that the rewards of satisfaction from assisting these kids are well worth their efforts.

For those of you who would wish to send contributions of hardware, software, firmware or other assistance, please mail it to:

Santuary, Inc.
1120 Fair St.
Eunice, LA 70535
1-318-546-0551

Thank you for your consideration in this matter.

Roger C. Bull
Gretna, LA

Say What?

I have a few comments on "Regression and Correlation" by C. Brian Honess (July 1981).

When you fit a set of points (X,Y) to a polynomial model (i.e. $Y^1 = Eb_i - x_i + a$), it is normally called "polynomial regression" and is distinct from *nonlinear regression*. In the latter, the nonlinearity refers to the relationships among the fitting parameters, not the linearity of the model itself. Thus, any function of the form $Y^1 = Ef_i(X)b_i + a$ is *linear* in the fitting parameters (a and b_i). A nonlinear model that is nonlinear in the parameters is any function that cannot be formulated in this manner. For example, $Y = b_ie^{-xb2}$ is nonlinear in the parameters. Note, however, that this particular function can be made linear through the use of a logarithmic transformation, e.g.: $\log Y = \log(b_ie^{-xb2}) = \log(b_1) - b_2\log(x)$.

Also, the method used to extract determinates is archaic. A much better method is to use Gauss-Jordan elimination to implement an upper triangular matrix transformation. This method is simpler, faster and easy to program. It also allows expansion to any size square matrix without rewriting the program, whereas the algorithm presented here is size dependent.

I am glad to see that your fine publication includes an occasional article that deals with mathematics. I hope this trend continues.

Bruce Douglass
Dept. of Physiology
University of South Dakota
Vermillion, SD

Subscript Reconsidered

Thank you very much for reviewing our Subedit/Subscript Word Processing software in the June issue. We appreciate the coverage and the encouragement you've given us, but regret that some of the inaccuracies and omissions in the article misled many of your readers and cost us a number of sales. We know this was inadvertant, and to mend some of the damage, we would like an opportunity to offer some clarification and updated information.

You indicated that Subedit/Subscript is written in Basic. However, Basic comprises less than 50 percent of that package (and only about a third of its successor, Newscript). The text formatting routines and the keyboard interface (Mininit) are in machine language. As a result, the system can keep up with *any* typist (2500 wpm, which is faster than the TRS-80 normally runs), and drive a 100 CPS printer at full speed most of the time. We used Basic for file I/O, storage management and printer tailoring; and machine language for speed. The combination is ideal: it makes it easy for us to distribute corrections, and easy for our customers to tailor Subscript to unusual needs. Given these speeds, there is little reason to convert the rest of the package to machine language. Since we encourage our customers to make backups, we saw no need to use machine language just to protect the code.

Of course, the original claim to fame of Subscript is its support for the Line Printer IV (Centronics 737): Subscript automatically performs right-justification of proportional font, allows inter-mixing of double- and single-width characters, performs underlining and does sub-scripts and super-scripts. It was the first TRS-80 Word Processor to offer these features, but the review article unfortunately didn't mention any of this support. We do thank you for describing how thoroughly we support

Continued on page 16

the MX-80, and wish to mention we also support the Diablo 1620, Selectric and other similar printers. We also were the first company to offer and deliver disk-based word processing for the TRS-80 Model III.

The review *did* identify a characteristic of Subedit that some people found annoying: the need to hit Enter after every 255 characters. I'm one of those people, and as the author, was in a position to do something about it! The result was Newscript, with a full-screen editor modelled after IBM'S Edgar and SPF, and which takes advantage of the unique abilities of the TRS-80. By coincidence, our first ad for Newscript was run in the same issue as your review article. I'm not sure if the ad overcame the effects of the article or was discredited by it . . . I only wish you had looked at your own advertising before running the article, or called us to verify its accuracy.

Newscript has all 50 of the formatting features of Subscript, plus some new ones. However, most of its enhancements are in the text entry and revision areas (Editing). The Mininit keyboard interface is built-in. The Full-Screen editor (written in machine language) scrolls, has a Whoops command, opens windows, moves words to the next line of the screen, allows the cursor to be placed anyplace on the screen, and permits insertion, deletion and overlaying of data directly at the cursor position. With its standard typeahead feature, Newscript can accept over 750 keystrokes per second (that's 7500 words per minute, which is probably fast enough for most of us) without ever losing a keystroke even at the end of a line. I mention all this mostly to lay to rest any notions some people may have about the slowness of Basic or the TRS-80: properly programmed, (as we have done), it's as fast as a small mainframe computer!

Chuck Tesler
Prosoft
North Hollywood, CA

Shell Sort Looped

It is said that each type of sort has its advantages and disadvantages. The Bubble sort is the slowest, however it takes the least amount of programming effort and space. The Shell-Metzner sort is one of the fastest and most practical sorts to use, but it takes a lot of program space.

I personally have found the Shell sort to be the best. I see no reason why it must take up so much space. In every book or magazine where I have seen a Shell sort listed, it is always in a longhand program style. I don't understand why no one has been creative enough to put the Shell-Metzner sort in For. . . Next loop form, but this routine took me a few short minutes to put together from the standard Shell sort flow chart.

Steven Graham
Forest Hills, NY

```
1000 '****** SHELL-METZNER SORT SUBROUTINE ******'
     A$( )---THE LIST TO BE SORTED
     NE------THE NUMBER OF ELEMENTS IN THE LIST
1010 M = NE
1020 M = INT(M/2):IF M = 0 THEN RETURN
1030 FOR I = 1 TO M : FOR J = I TO NE – M;1 – M STEP M : FOR K = J + M TO NE – M + 1 STEP M
1040 IF A$(J)>A$(K) THEN T$ = A$(J):A$(J) = A$(K):A$(K) = T$
1050 NEXT K,J,I:GOTO 1020
```

Program Listing

Memory Lapse

I have reached the point of pure sickness at Radio Shack. They try to get you to buy their software and hardware, but when you upgrade to disk and go to transfer the programs you have such as Pyramid, Invasion Force, and Eliza to disk they tell you you need the memory addresses, and even when you need to relocate them Radio Shack doesn't lift a finger to help you.

We trusted Radio Shack with our hard-earned money for their computer, and they slapped us in the face with the used car salesman "Trust me" routine. It's time to remind Radio Shack that we are paying for their goods, and without us they'd crumble.

Robert Rose
Oakland Park, FL

Just What He Needed

Mr. Rose's complaint is a valid one; I'm sorry we haven't taken care of this one sooner.
Here's a list of the necessary addresses:

	Start	End	Entry
Micromovie—(MOVIE)	4300	4CFF	4300
Eliza—(ELIZA)	5000	7800	5000
Talking Eliza—(ELIZA)	4600	7C00	4600
Microchess—(CHESS)	cannot load on disk, special loading format.		
Micromerquee—(MARQ)	4A00	4FFF	4B00
Invasion Force—(INVADE)	5000	7100	5000
Flying Saucer—(SAUCER)	42EA	4FFA	42EA
Editor Assembler—(EDTASM)	4300	5D40	468A
TBUG L = III—(TBUG)	4380	4824	43A0
In-Memory Initialization L-II—(INITLZ)	4380	478C	4380
Retrieval L-II—(RETREV)	4380	4B8C	4380
Sort L-II—(SORT)	4380	46DE	4380
Checkers 80—(CKRS80)	5000	7700	5000
Program Conversion—(CONV)	4AF6	4FD7	4AF6
Data Conversion—(DCONV)	4300	4525	4300
Micromusic—(MUSIC)	4300	4970	4300
Renumbering Program—(RENUM)	7C4C	7FC6	7C4C
Term—(TERM)	5000	50BF	5000
Pyramid—(PYRMD)	4300	7FFE	4300
Haunted House—(HAUNT)	42E9	4FFF	42E9
Scripsit Cassette—(SCRIPS)	4300	69C5	4300
Upper & Lower Case Driver Basic—(ULCBAS)	6C00	7015	6C00
Disk—(ULCDVR)	7000	73FF	7000

Bill Walters
Consumer Information Manager
Tandy/Radio Shack
Fort Worth, TX

OP Code Change

After reading Brian Cameron's article "Undocumented Instructions" (July 1981), I began looking through op code tables out of curiosity to verify some of the codes. They are indeed undocumented. After further researching, I found that Mr. Cameron's title of 'DUPINC' for the instruction to multiply by two and increment is actually 'SLL' (Shift Left, Logical). The other op codes he names appear correct.

Just for the record, I'm 16 years old.

P. Griffith
Tampa, FL

Reversing Graphics

Regarding your article on graphics codes ("Unlocking the Graphics Code," June 1981), there is third application. Through a little math, a graphics pattern can be reversed—exchange black for white and vice versa. To see how this works (Model I and Model III), type in the following:

```
10 CLS: for X = 128 to 191: PRINT @ o,x;: POKE 15365,
x 20 POKE 15370, 191 – (PEEK(15365) – 128): FORY = 1
to 300: NEXTY,X.
```

The contents of 15365 are reversed and POKEd to 15370. To reverse all graphics on the screen, use the subroutine: 10000

Continued on page 20

✓See List of Advertisers on page 386

80 DEBUg

Broker Botches

I have found an error in one of the listings contained in my article "The Software Broker" which appeared in the June 1981 issue of *80 Microcomputing*. In the 'MOVING/AVE' program (Listing 4), line 200 now reads: 200 A = Q + 1. This should be changed to: 200 Q = Q + 1:AV = 0.

I am sorry if this has created a problem for any of your readers.

John Harper
Rte. #1 Box 252
Lawrenceburg, IN 47025

Hi- Res Help

The April 1981 issue of *80 Microcomputing* had one error in the "High Resolution Video Interface" article. The headlines should have read 192 by 96, not 192 by 26. Also, the schematic shows a 5257 or 4041 as the memory chips. The 4041 is getting difficult to find so use the TMS-4044, MM-5257 or INTEL 2147.

Several recurrent questions have come up that I would like to take the opportunity to answer. The size of the HR-1 board is 3 by 7 and 7/8 inches. Second, the size of the smallest dot is equivalent to the size of the period on the TRS-80 CRT and 18,432 of them can be defined anywhere on the screen. The normal TRS-80 graphics can define only 6144 individual dots.

Third, is the complete (assembled) unit available? At this time, only the circuit board is available. However, there is a company in Augusta, GA, Cardin and Associates, that is considering supplying kits and fully assembled boards. Judicious buying should allow you to gather all the parts (including circuit board) for \$75 to \$80. The six memory chips are the most expensive items. Some distributors want up to \$16 each for these ICs, but they are available for less than \$5 each from

Continued on page 22

Dirprog Update

In the August 1981 issue of *80 Microcomputing*, we published an article called Dirprog by Jack Egbert. Unfortunately, we ran an older version of the program. The program listing here is the updated (better) version of Dirprog. We hope this eliminates any confusion publishing the older version may have caused:

```
10 '              ***   DIRPROG   ***
15 '            ========================
20 ' A PROGRAM TO MAINTAIN A DISK DIRECTORY INDEX FILE
30 :
40 ' VERIFY GSF48/OBJ IS IN MEMORY, INITIALIZE IT
50 IF PEEK(&HFE80)<>205 OR PEEK(&HFFA0)<>26 CLS:PRINT@192,
   " **  GSF48/OBJ  NOT LOADED  **

      LOAD GSF48/OBJ --- AND THEN DIRPROG":PRINT@900,:END
60 DEFUSR=&HFE80:CLEAR:POKE -1,0
70  A$="":DN$="":E$="":SW$="":TI$="":TP$="":WF=0 ' WRITE FLAG=WF
80 CLEAR 16000:DIM PN$(952),DN$(952)
90 DEFINT J,X :XX=0
100 CLS:PRINT@156,"    MENU":PRINT
110 PRINT,"< R > -- READ FILE FROM DISK"
120 PRINT,"< W > -- WRITE INDEX FILE TO DISK"
130 PRINT,"< A > -- ADD TO FILE"
140 PRINT,"< D > -- DELETE"
150 PRINT,"< S > -- SORT AND OPTIONALLY PRINT-OUT FILE"
160 PRINT,"< F > -- FIND A PROGRAM"
170 PRINT,"< L > -- LIST BY DISK IDENTIFIER"
180 PRINT,"< V > -- VIEW A DISK'S DIRECTORY"
190 PRINT,"< E > -- EXIT THE PROGRAM SAVING INDEX"
200 PRINT:PRINT"INPUT YOUR SELECTION . . . .":GOSUB 1600
210 IF A$="R" GOSUB 930:GOTO 100
220 IF A$="W" GOSUB 830:GOTO 100
230 IF A$="A" GOSUB 350:GOTO 100
240 IF A$="D" GOSUB 580:GOTO 100
250 IF A$="S" GOSUB 1050:GOTO 100
260 IF A$="F" GOSUB 670:GOTO 100
270 IF A$="L" GOSUB 1320:GOTO 100
280 IF A$="V" GOSUB 1400:GOTO 100
290 IF A$="E" GOSUB 850:CLS:END
300 GOTO 200
310 '***   SUBROUTINES   ***
320 '
330 '**  SUB  **           ADD DISK TO FILE
340 '   ===================================
350 CLS:PRINT:PRINT,"INSERT DISK":PRINT:XX=XX+1
360 PRINT"ENTER DISK NO. - HIT (TAB) - THEN ENTER TITLE OF DISK"
370 PRINT:LINEINPUT DN$(XX):WF=1
380 CMD"DIR :1"  ' ** USE - CMD"DIR :0" - FOR ONLY ONE DRIVE
390 DN$=LEFT$(DN$(XX),4):PN$(XX)="!"
400 SP=15488 :Z=0 :PN$=""
410 XX=XX+1:DN$(XX)=DN$:GOSUB 500
420 IF PEEK(15488)=32 THEN PN$(XX)="# "+MID$(DN$(XX-1),9,11):GOTO460
430 IF Z=-1 THEN XX=XX-1:GOTO 460
440 IF Z/3 = INT(Z/3) THEN SP=SP+24 ELSE SP=SP+20
450 GOTO 410
460 PRINT:PRINT"TO ENTER ANOTHER DISK -- HIT < A >"
470 GOSUB 1450
480 IF A$="A" GOTO 350
490 IF A$=CHR$(13) THEN RETURN ELSE GOTO 460
500 IF CHR$(PEEK(SP))=" " THEN Z=-1:RETURN
510 LO=PEEK(16598):HI=PEEK(16599)
520 FOR X=0TO11:PN$=PN$+CHR$(PEEK(SP+X)):IF CHR$(PEEK(SP+X))=" " THEN X=11
530 NEXT X:POKE 16598,LO:POKE 16599,HI
540 PN$(XX)=PN$:PN$="":Z=Z+1:RETURN
550 :
560 ' **  SUB  **          DELETE A DISK FROM INDEX
570 '    ==========================================
580 CLS:PRINT
590 INPUT"WHAT IS THE NO. OF THE DISK YOU WANT TO DELETE ";DN$
600 WF=1:CLS:PRINT@340,"WORKING . . . ."
610 FOR J=1 TO XX
620 IF LEFT$(DN$(J),4)=LEFT$(DN$,4) THEN DN$(J)="ZZ":PN$(J)="ZZ"
630 NEXT J
640 CLS:PRINT@330,"DISK NO. -- ";DN$;" -- IS DELETED"
650 GOTO 1450
660 :
670 ' **  SUB  **          TO FIND A PROGRAM
680 '    ==================================
690 CLS:PRINT@192,"ENTER TITLE OF PROGRAM TO SEARCH FOR . . ."
700 PRINT:LINEINPUT TI$
```

Continued on page 20

FOR X = 15360 to 16383: POKE X, 191 − (PEEK (X) − 128): NEXT : RETURN. If the screen contains some alphanumeric characters, add "IF PEEK (X)<128 THEN NEXT ELSE" before the word "POKE". When graphics are stored in strings (in A$), reversal would be:

```
1000 B$:"": FORX= to LEN (A$): B$=B$+CHR$
(191 − (ASC(MID$(A,X,1)) − 128)): NEXT.
```

Also similar routines could be used to exchange characters for graphics and vice versa, exchange uppercase for lowercase and vice versa, or to use INKEY$ to print graphics.

Marc Brumlik
Computer Marketing
Radio Shack
Groveport, OH 43125

On Time Fix

In reference to the article "Soft Tach" (June 1981), I would like to point out that the program may hang up if the disk motor on time is less than the time required for 12 revolutions. The motor on time is nominally 2.6 seconds and the time required for 12 revolutions is 2.4 seconds at 300 rpm.

However, due to component tolerances, the on time may be shorter, and also the first revolution of the disk will be slower than 300 rpm.

An easy fix is to reduce the total number of revolutions. For seven revolutions change the 24 in line 90 to 14, and change the following lines:

```
230 FOR N = 1 TO 5
390 AV = RT/5
```

This will give an average of five disk revolutions.

David Cheney
Lachine, Quebec
Canada

An A for Alpha

I recently received excellent service from the Alpha Product Company.

I placed a telephone order and had the merchandise in a few days. It wasn't quite the right thing, unfortunately, so I returned the item. With equal speed, checks arrived for the refund and shipping costs. I was pleased by the service, speed and honesty. I will certainly look to Alpha for my future needs, and I'll recommend them to others without hesitation.

Robert A. Martin
Albuquerque, NM

80 DEBUG

Program continued

```
710 CLS:PRINT@335,"SEARCHING FOR :  ";TI$:PRINT:PRINT
720 PRINTTAB(5)"TITLE";TAB(20)"DISK":PRINTSTRING$(34,"=")
730 FOR J=1 TO XX
740 Q=INSTR(PN$(J),TI$) :IF Q=0 THEN 760
750 IF TI$=MID$(PN$(J),Q,LEN(TI$)) THEN PRINTTAB(5)PN$(J);TAB(20)LEFT$(DN$(J),5)
760 NEXT J:PRINT:PRINT
770 IF XX=0 THEN PRINT"**  DISK INDEX FILE NOT LOADED !!  **":PRINT@900,:END
780 PRINT"THAT IS ALL !!!  ---  HIT <ENTER> FOR MENU . . ."
790 GOSUB 1600:RETURN
800 :
810 ' **  SUB  **        WRITE INDEX FILE TO DISK
820 '     ====================================
830 CLS:PRINT:PRINT"MAKE PREPARATIONS FOR OUTPUT TO DISK . . ."
840 PRINT:PRINT,"WHEN READY -- HIT <ENTER> . . .":GOSUB 1600
850 IF WF=0 RETURN ELSE OPEN"O",1,"DISKINDX"
860 CLS:PRINT@340,"WRITING FILE TO DISK . . ."
870 PRINT#1,XX:FORX=1 TO XX:PRINT#1,DN$(X);",";PN$(X):NEXT X
880 CLOSE:CLS:PRINT@335,"TRANSFER TO DISK COMPLETED"
890 WF=0:GOTO 1450
900 :
910 ' **  SUB  **        READ INDEX FILE FROM DISK
920 '     =====================================
930 CLS:PRINT@128,"MAKE PREPARATIONS TO READ FILE FROM DISK"
940 PRINT:PRINT"WHEN READY -- HIT <ENTER> . . .":GOSUB 1600
950 CLS:PRINT@340,"INPUTTING FILE FROM DISK . . . ."
960 OPEN"I",1,"DISKINDX"
970 INPUT#1,XX:FOR X=1 TO XX:INPUT#1,DN$(X),PN$(X)
980 IF VAL(PN$(X)) > 99 PN$(X)="   "+PN$(X)
990 NEXT X:CLOSE:WF=0
1000 CLS:PRINT@335,"INPUT OF FILE FROM DISK COMPLETED."
1010 GOTO 1450
1020 :
1030 ' **  SUB  **        SORT BY PROGRAM OR DISK NO.
1040 '     ========================================
1050 CLS:PRINT@192,"SORT THE INDEX FILE BY -- :

        < 1 > -- DISK NO.
        < 2 > -- PROGRAM TITLE"
1060 PRINT:PRINT"PLEASE MAKE YOUR SELECTION . . .":GOSUB 1600
1070 MS=VAL(A$):IF MS<1ORMS>2 THEN GOTO 1050
1080 IF MS=1 GOSUB 1490 ELSE GOSUB 1510
1090 CLS:PRINT@192,,"< R > -- RETURN TO MENU
                < L > -- LIST THE COMPLETE FILE

PLEASE MAKE YOUR SELECTION . . . ."
1100 GOSUB 1600:IF A$="R" THEN RETURN
1110 IF A$<>"L" GOTO 1090
1120 :
1130 ' **  SUB  **       OPTIONAL PRINT-OUT
1140 '     ================================
1150 CLS:PRINT@192,"OUTPUT TO PRINTER  ( Y  OR  N ) ?"
1160 GOSUB 1600:N=0:IF A$="Y" THEN LP=1 ELSE LP=0
1170 FOR X=1 TO XX
1180 L$=LEFT$(PN$(X),1):L5$=LEFT$(PN$(X+54),1):IF MS=1 GOTO 1200

1190 IF L$="!" GOTO 1280
1200 IF L$="!"OR L$="" PRINT"";ELSEPRINT PN$(X);TAB(13)STRING$(6
,"-");"  ";LEFT$(DN$(X),5);
1210 IF L5$="!" OR L5$="" OR X+54 > XX PRINT"" ELSE PRINTTAB(36)
PN$(X+54);TAB(49)STRING$(6,"-");"  ";LEFT$(DN$(X+54),5) ELSE PRI
NT""
1220 IF LP=1 AND PEEK(14312)<>63 THEN CLS:PRINT@330,"PRINTER NOT
 READY !":GOTO 1450
1230 IF LP=0 THEN GOTO 1270
1240 IF L$="!" OR L$="" LPRINT" ";ELSE LPRINT PN$(X);TAB(13)STRI
NG$(6,"-");"  ";LEFT$(DN$(X),5);
1250 IF L5$="" OR L5$="!" LPRINT" ":GOTO 1270
1260 IF X+54 =< XX LPRINTTAB(45)PN$(X+54);TAB(58)STRING$(6,"-");
"  ";LEFT$(DN$(X+54),5) ELSE LPRINT" "
1270 N=N+1:IF N=>54 THEN N=0:X=X+54:IF LP=1 LPRINTSTRING$(12,13)

1280 NEXT X:GOTO 1450
1290 :
1300 ' **  SUB  **      LIST DISK ID'S IN INDEX FILE
1310 '     ========================================
1320 CLS:FOR X=1 TO XX
1330 IF PN$(X)="!" THEN PRINT DN$(X) ELSE 1350
1340 IF PEEK(14312)=63 LPRINT DN$(X)
1350 NEXT X:PRINT
1360 PRINT,"END OF FILE -- TO RETURN TO MENU, HIT <ENTER>":GOSUB
 1600:RETURN
1370 :
1380 ' **  SUB  **      VIEW A DISK'S DIRECTORY
1390 '     ===================================
1400 CLS:PRINT@320,"";:INPUT"ENTER DISK NO. YOU WISH TO REVIEW "
;DR$
```

Program continues

80 DEBUg

companies advertising in the major computer magazines—keep looking! The May 1981 *80 Microcomputing* does not state on the parts list that all TTL devices must be of the low power Schottky type (74LSXX series). If you do not use low power devices the TRS-80 power supply may not be able to cope.

Next, whether your system has 48K of memory or not, disregard the section pertaining to the memory disable option. Instead use the technique shown in Fig. 1.

Paul C. Fowler, Jr.
Enable Electronics
2103 Charlton Lane
Radford, VA 24141

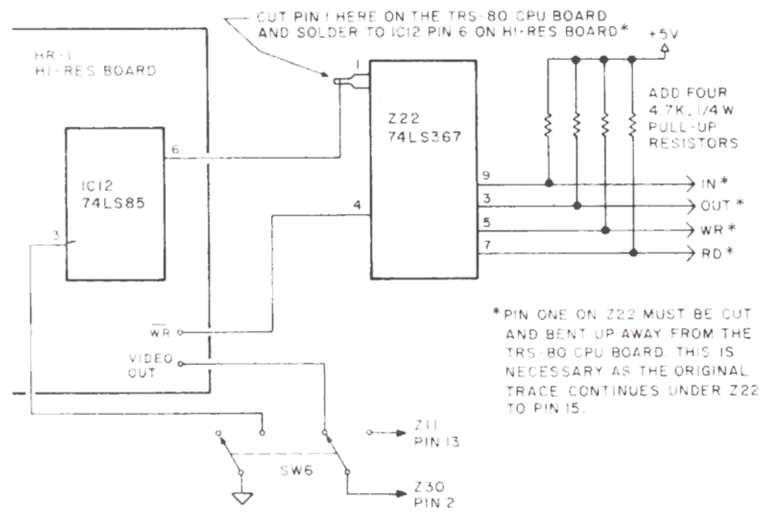


Fig. 1

High Addresses

An apology is due readers with 16K systems who tried the demo program in my article "Program Chaining and Local Variable Definitions in Basic" (June 1981 p. 255). Despite considerable thought given the selection of addresses for the start of variable storage, I failed to heed the advice in my own article and set one of them too high. Consequently when run in a 16K system the stack overwrites part of that variable storage area and the program bombed. It runs OK in systems with more memory.

A quick fix is to change the number 32500 to 30500 the four places it appears in the listing (Lines 120, 170, 230 and 1010). The program will then run on any size system.

Hal Brown
643 W. Valley Forge Road
King of Prussia, PA 19406

Program continued

```
1410 CLS:PRINT@128,"DIRECTORY FOR DISK NO.  ";DR$:PRINT
1420 FOR J=1 TO XX
1430 IF LEFT$(DN$(J),4)=DR$ AND PN$(J)<>"1" THENPRINT PN$(J),
1440 NEXT J:PRINT:PRINT
1450 PRINT:PRINT,"TO RETURN TO MENU -- HIT <ENTER>":GOSUB 1600:R
ETURN
1460 :
1470 ' **  SUB  **     GSF SORT ROUTINE
1480 '     ===============================
1490 SP$="+DN$,+PN$"
1500 GOTO 1520
1510 SP$="+PN$,+DN$"
1520 CLS:PRINT@330,"SORTING ---- ONE MOMENT PLEASE"
1530 IF XX=0 THEN PRINT@450,"**  DISK INDEX FILE NOT ENTERED !!
 **":GOSUB 1450:GOTO 100
1540 I=USR(17) OR USR(VARPTR(SP$)) OR USR(0) OR USR(XX)
1550 IF I <> 0 THEN PRINT"SORT ERROR":GOTO 1550
1560 FOR X=1 TO XX
1570 IF DN$(X)="ZZ" THEN XX=X-1:RETURN
1580 NEXT X:RETURN
1590 :
1600 A$=INKEY$:IF A$="" THEN 1600 ELSE RETURN
```

Program Listing

Everyman's Debug

I typed in "Everyman's Mod II Word Processor (July 1981) on my Model III. Everything works well with the following exceptions: To fix faulty video display during edit insert mode, change the end of line 1080 to read:—:PRINT @ C + 1, B$;A$ + R$

Since Model III doesn't have tab or backspace keys, and does not recognize the up arrow, change line 180 to read:

```
180 IF A = 62 THEN 430 ELSE IF A = 91 THEN 330
ELSE IF A = 60 THEN 310
```

This lets you use "<" for the left arrow, ">" for the right arrow "Clear" for the down arrow, and the up arrow as in the text. If moving the line to the right chops off the last character, delete the + from line 1200.

If you use an Epson MX80 line printer, lines 1661 and 1662 will have to be changed to accommodate its char/inch and lines/inch format.

```
1661 IF G = 5 THEN W = 14 ELSE IF G = 10 THEN
W = 18 ELSE IF G = 16.5 THEN W = 15
1662 IF H = 6 THEN T = 50 ELSE IF H = 8 THEN
T = 48 ELSE IF H = 10 THEN T = 49
```

Howard Potvin
2527 S. Los Padres Drive
Rowland Heights, CA 91745

DEBUG

The Real Thing

What! Another correction? In August's *80 Input*, I commented that Frank DiNunzio's joysticks might cause bus contention if used during a CLOAD. That was true, but between my drawing errors compounded by drafting errors the fix that was described would end up being worse than no fix at all. Fig. 1 and Fig. 2 are the real thing.

Dennis Kitsz
Technical Contributing Editor
Roxbury, VT

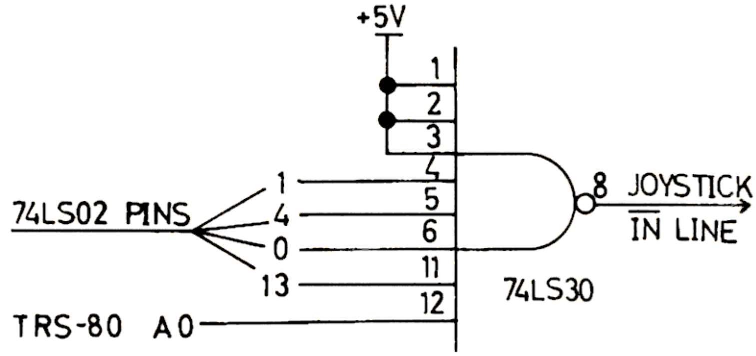


Fig. 1

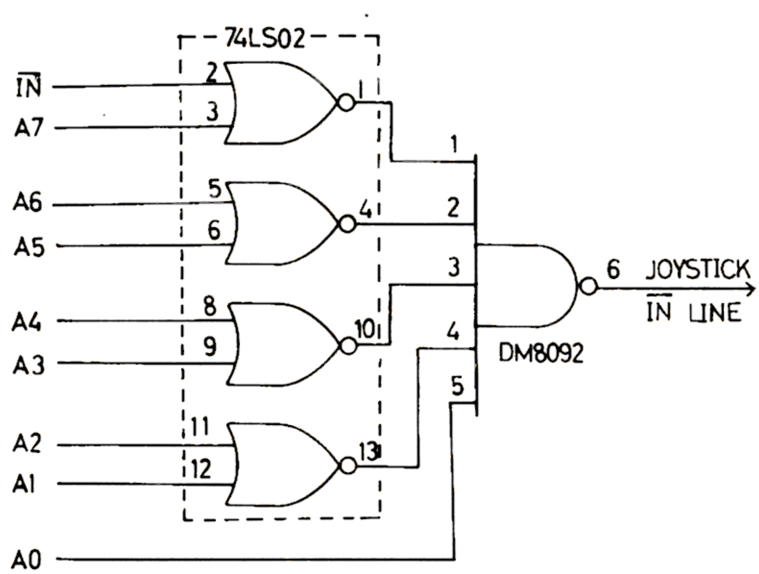


Fig. 2

Joysticks Damage Hardware

As Technical Contributing Editor for *80 Microcomputing,* I am often called on to review hardware manuscripts before their publication. Frank DiNunzio's article on joysticks (June 1981) was such a case. I contacted Mr. DiNunzio before the article was published pointing out my concerns, to which I received a response similar to the one printed in September's Input. I reviewed the article again, and came to the same conclusions. Since he did not wish to incorporate a protective port-decoding circuit in the joystick design and since I believed this would be a very popular project, I prepared a postscript to the article.

For a time, the Keystone Kops syndrome set in. Due to magazine madness, the postscript disappeared and the article was published without it. I sketched out a hasty replacement; unfortunately, I made a mistake in the drawing, later compounded by a drafting error, which completely garbled the so-called improvement (please see this month's Debug). Mr. DiNunzio's rather harsh response to my suggestions then appeared.

If this were a minor theoretical disagreement, I would shelve any response in order to avoid an unseemly and typically mundane war of letters to the editor. But the user's investment in a computer is considerable, so I want to answer Mr. DiNunzio's letter point by point.

First, I understood that the joysticks were meant to be used only with "a suitable program to draw or play games." However, it is unwise and chancy to plug something in when the computer is on, and user manuals for all computer peripherals—including Radio Shack's own—warn you to turn your equipment off before connecting them. Since it is important that any peripheral be plugged in when the power is off, Frank's joysticks would be in *and on* when CLOADing such a suitable program.

That brings up a related comment I probably should also have made earlier. Using the TRS-80 bus connector—pins 37 and 29—to complete the circuit that powers the joysticks puts the computer in jeopardy. Plugging the board in on an angle can run a hazardous 6 volts through other components of the system. So put a switch on the joysticks, too.

Second point: Frank says "INP(1) will not open the cassette port; INP(225) does that." Absolutely right. But the converse is not the case, since opening the cassette port will also open the joysticks. So will input from 240,241,242,243,244,245, 246 and 247 (Exatron Stringy Floppy), input from 232, 233, 234 and 235 (RS-232 Interface), input from 208 and 209 (Microconnection), input from 127 (Micromouth), and so on. I save the chain programs of ESF, and it cannot be plugged or disabled without wiping out a loaded program or program sequence. And removing the RS-232 unit means disconnecting the expansion interface (killing a disk system) or opening the cover and removing the RS-232 board before proceeding.

Mr. DiNunzio's statement does bring up an interesting question: Why would INP(255) cause the joysticks to open while on the other hand INP(1) does not cause the cassette port to open? That is because INP(1) is a hardware illusion created by software; as I stated in my first comment, *any* INP statement from INP(0) to INP(255) will trigger these joysticks. Mr. DiNunzio uses INP(1) simply because specifying that port number will avoid simultaneously triggering the cassette input circuitry.

Third point: "Hex inverters . . . pull some of the data lines down to ground potential; this is also done by the keyboard and all other input devices, and is not harmful to the computer." That's true, but all the devices he mentions pull data lines down *only* when specifically asked to do so by port or address number. Frank's joysticks pull data lines to ground even when not asked to by proper port number, and that is harmful to the computer. Why is it harmful? Because digital electronic devices can have three states: on, off, and a third state of electronic invisibility. To avoid electronic chaos, only one device may be permitted to respond to a central processing unit's request for information. In cases where bus conflict happens accidentally (which will never occur in a properly designed circuit), the low (ground, or "zero") signal takes precdence. But while it is low, it is also causing additional current to flow through any circuits which may be in a high (5 volts, or "one") state. That's what the third state was designed to avoid, which should have been used here. The unorthodox method used by Mr. DiNunzio to turn on the 74LS368's is a partial attempt to provide this invisibility, though 74LS366's (with two enable lines) would have been a better choice in some ways. Though unusual, his method can extend battery life.

To provide signals only when needed, then, every device connected to the computer should be decoded; that is, it should have a specific address or port integrated with it. In that way, only one device is "visible" to the CPU, and no peripherals or memory compete with each other. These joysticks do in fact compete for attention, may respond unexpectedly, and (I repeat) *may eventually damage the computer.*

Fourth point: When I was reprimanded that "241 is the lowest data number the circuit develops," I checked the original article once again. Actually, the lowest data number produced is 240, as Frank points out in his abstract numbers; rather, they work in digital signals whose actions can be made to simulate decimal numbers. Thus, though 240 is not a decimal zero, that is irrelevant; it is binary 11110000—which means four of the eight independent data lines *are* pulled to zero, and can conflict with active computer data during other input. Readers can take a lesson here in the dangers of thinking in decimal numbers when binary concepts are being implemented; merely because 240 is not zero, it does not follow that the binary condition it represents has no components which *are* zero. The possible configurations of Frank's joysticks include several zeros in the lower four lines: 0000, 0001, 0010, 0011, 0100, 0110, 1000, 1001, 1100, and 1111. Nine of the ten combinations present bus contention problems.

Fifth point: Frank states that an AA cell "in theory . . . is 1.5 volts, but in reality it provides somewhat less voltage." In reality, depending on how new it is, an AA cell can provide *more* than 1.5 volts (as well as less), even under the minimal load demanded by the joysticks. However, I will grant his point that most of the time the voltage will be less than 6 volts. Nevertheless, fairly new alkaline cells will still provide more than 5.25 volts, the maximum operating supply voltage. Here are excerpts from the manufacturer's data sheet on the same integrated circuits specified by Mr. DiNunzio for this project:

> "Operating Conditions. Supply Voltage: (min) 4.75 (nominal) 5.00 (max) 5.25. Absolute Maximum Ratings. Supply Voltage: 5.00. Note: Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated . . . is not implied."

For greatest reliability, I would recommend readers use the simple regulator circuit I suggested (*80 Input,* August 1981, page 18) with *five* AA cells rather than four; this will prevent the regulated voltage from dropping below 5 volts too soon, giving the appearance that the batteries have gone dead early.

To Mr. DiNunzio's final provocations, I decline to respond. I will say that even if readers do decide to follow his design as printed, at least this exchange of correspondence and commentary should illuminate how different designers approach a task, and how hardware reliability can be increased. As I stated in my original letter, I did not intend to be hard on Mr. DiNunzio; I'm sorry he was offended, but I hope readers know that they are my first concern. It is they who have to face the consequences of what they build.

Dennis Bathory Kitsz
Technical Contributing Editor
Roxbury, VT

'Know-it-All' Goof

The program "Know-it-All" (May 1981, p. 288) is a very useful program but is not complete in every detail. First, there is an error in line 65035. The items between the brackets should read (65536-U).

Second, if there are any alpha data lines these will be decoded as a register and to eliminate this, insert OR T = 136 after T = 0 in line 65055.

Third, in program statements such as ON X GOTO 100,200,300, line referencing program only picks up the first line 100 and not lines 200 and 300. To overcome this, type in place of line 65150:

```
65150 NEXT:RE = 0:C$ = ""GOSUB650:IFT =
OTHEN65045
65152 IFT = 320RT = 44THEN65130ELSEGOTO65050
```

With these additions the program becomes very useful in debugging lengthy programs.

Brian Heywood, chief engineer
Taranki Electric-Power Board
Eltham, Taranaki
New Zealand

Retrieving Block Cursor

I read the article on "Block that Cursor" by Ron Balewski in the April 1981 issue and noticed that if I tried to relocate the program to a location high enough to use my 48K of RAM I would get an "OV" error from BASIC. Evidently the problem is in the integer arithmetic that is used by the PEEK and POKE statements. While the Z-80 can handle addresses greater than 32767, the TRS-80 integer arithmetic cannot.

Fortunately, there is a way out. You can use the block cursor and your additional memory. It turns out that if you subtract 32768 from the 16K POKE locations the TRS-80 two's complement arithmetic will come up with the appropriate addresses for the POKEing. This is done on line 30 in the program below. Of course, the new memory size will be 65402, and you must change the location that will be referenced by Basic, which is done as follows:

```
5 'BLOCK CURSOR PROGRAM FOR 48K RAM    SET
   MEMORY SIZE TO 65402
10 FOR K = 32635 TO 32654
20 READ X
30 POKE K - 32768,X
40 NEXT K
50 DATA 245,197,205,88,4,237,75,32,64,10,254,95,32,3,
   62,143,2,193,241,201
60 POKE 16414,123
70 POKE 16415,255
80 END
```

William R. Patterson
Somerdale, NJ

Please call (213) 883-8594.

Our Epson prices are so low, we're not allowed to print them.

EPSON MX-80 ....$CALL

80 cps/9x9 matrix/Lower case with true descenders/Bi-directional & Logic seeking/Adjustable tractor/Expanded printing/Block graphics/Forms control/Compressed printing/Double-strike printing/Correspondence quality/Emphasized printing mode/Standard parallel interface

EPSON MX-80 F/T ......$CALL

Same features as the MX-80 plus Friction Feed. Adjustable removable tractor is standard for ease of handling forms and single sheets.

EPSON MX-100 F/T .....$CALL

Same features as the MX-80 & MX-80 F/T but on 15½ inch carriage for printing 132 columns with standard 10 cpi font or 232 columns in the compressed character font. The MX-100 is complete with Dot Resolution Graphics

EPSON INTERFACES & OPTIONS

TRS-80 MODEL I, III CABLE	$ 30	SERIAL INTERFACE (2K BUFFER)	$ 149
TRS-80 MODEL I Keyboard Interface	$ 95	SERIAL CABLE Male to Male	$ 30
TRS-80 MODEL II CABLE	$ 30	DOT RESOLUTION GRAPHICS	$ 90
APPLE INTERFACE & CABLE	$ 100	REPLACEMENT RIBBON	$ 13
IEEE 488 INTERFACE	$ 60	REPLACEMENT PRINT HEAD (Quiet type)	$ 40
SERIAL INTERFACE	$ 70	EPSON SERVICE MANUAL	$ 40

✓69

Alpha Byte STORES

We built a reputation on our prices and your satisfaction.

We guarantee everything we sell for 30 days. If anything is wrong, just return the item and we'll make it right. And, of course, we'll pay the shipping charges.

We accept Visa and Master Card on all orders. COD accepted up to $300.00. We also accept school purchase orders.

Please add $2.00 for standard UPS shipping and handling on orders under 50 pounds, delivered in the continental U.S. Call us for shipping charges on items that weigh more than 50 pounds. Foreign, FPO and APO orders please add 15% for shipping. California residents add 6% sales tax.

31245 LA BAYA DRIVE, WESTLAKE VILLAGE, CALIFORNIA 91362

✓See List of Advertisers on page 386

80 REVIEWS

edited by Michael Nadeau

"Mr. Baker has some unique ideas compared to others involved in software production."

Trakcess
Roxton Baker
The Alternate Source
Lansing, MI
$24.95

by Ronald Bobo

After successfully making backup copies of Acorn's "Pinball" and Microsoft's "Adventure" (both of which are on copy-protected diskettes), I next turned my attention to those annoying security diskettes that accompany programs from The Bottom Shelf.

Mr. Baker has some unique ideas compared to others involved in software production. To quote from the Trakcess manual, "Congratulations! You have just purchased *the most powerful* TRS-80 disk access utility yet written. Perhaps you were able to share the cost with a few friends, but even if not I hope you will find Trakcess worth the price."

Baker credits William Barden's *Disk Interfacing Guide for the TRS-80* as inspiration for this program and recommends it to help you use Trakcess efficiently. The non-expert should also have a copy of the data sheet on the 1771 floppy disk controller chip. A copy can be found in the Expansion Interface service manual or may be obtained from Western Digital. National Semiconductor, a second source for the chip, also has a sheet available.

Trakcess requires a 48K TRS-80 Model I. The (C)opy and (D)uplicate commands require two disk drives.

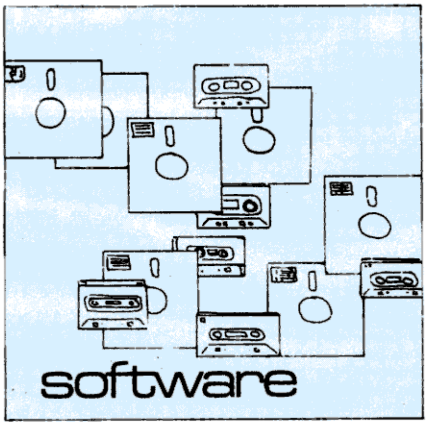
The program is in two parts on the disk—Trakcess, a Basic program, and Trakcess/CMD a machine-language routine. Typing 'Trakcess' and pushing Enter will load the machine-language section, and from there on prompts are given.

Printer output is available, but a printer is not necessary.

Trakcess contains all its own disk I/O routines. If the program stops on an error, however, a DOS disk should be inserted in drive 0 before continuing, to prevent possible system hangup.

Everything is menu-driven. Disk-related commands will not work until a drive is selected and activated. Upon selection, you will be asked for the head position of the drive. Press Enter; the default value is zero, and the head will be positioned at that track.

Typing I steps the head of the selected drive in one track (toward the center of the disk) and O steps it out. These two keys repeat if held down. Trakcess is programmed for thirty-five tracks, but instructions are furnished for changing this if necessary.



G followed by a decimal number allows quick positioning of the head at any desired track.

R and W allow reading into a specified block of memory, or writing from it. It is feasible to modify the data before writing back to disk, making the reconstruction of a damaged sector possible. All sector writes are verified by rereading, so data transfers are reliable.

T and P (Take and Put) allow the taking from or putting to disk of a whole track (about 3120 bytes) in one operation. A track may be read in, then scanned with Trakcess' editor to reveal everything on the track.

B (Build) is a very powerful command. It allows you to tailor a track to your own specifications, with no restrictions other than overall length. As many as eighty different sectors per track may be specified, of different lengths and types, with any names. Once the track has been created it may be edited in memory before writing it to disk, with the P command. For example, you might zero out the DAM (Data Address Mark) for a short sector, thereby generating a false sector ID. Or you may remove the ID CRC to get a false ID pack or the sector CRC for an always-bad sector. Make your own protected disks, anyone?

Typing S causes the current track to be scanned for useable sectors. All the important information (track number, sector number, sector length, whether or not a sector is IBM format, and the data address mark) will be determined and displayed on the screen or printer. This process is not fast; thirty seconds may be required to scan a track. If the track has any false sectors, this will be noted. L (Locate) scans the full disk, tells you which tracks have sectors, then offers you the option of a full report.

C will search the current track, build a matching format track in memory, then write it out to a target disk. Subsequently, it will transfer (and verify) all the sectors. If any sectors are damaged or of undeterminable length, you will have to specify one. Unless you know what it should be, do a track read and look at the sector in memory. Try the next larger multiple of sixteen bytes than the sector's apparent length. For this and the D command, you must have two drives—you cannot copy to the active drive.

D is essentially the C command repeated for each track. Most disks, whether protected or not, can be duplicated in about thirteen minutes. The disk you're copying need not be formatted, but it's a good idea to format it to make sure there are no bad tracks. It may then be bulk-erased before copying.

E is a scrolling editor utility. Memory may be edited, or memory between specified addresses may be filled with a specified byte. The arrow keys move the cursor, and scrolling is accomplished with shifted up and down arrows. Memory may be displayed in either hex or ASCII. In either mode, whatever you type will be put into memory at the cursor location.

F calculates the two CRC bytes for any block of code in memory, or for any bytes

THE ALPHA I/O SYSTEM

a complete failure?

THE INSIDE STORY

It happened 3 years ago, when our President made a decision. At the time we specialized in custom analog and digital circuit design. The decision was to attempt to develop a line of standard interface hardware for the emerging microcomputers. At the time (1977) we had to decide which of the new machines could become the "industry standard" of the low cost micros.

Despite a few aggravating but minor deficiencies, the TRS-80 seemed tohave the most chance of success and it had the best price/performance ratio. Also, with some imagination, their large sales organization could become the largest service network in the world, a reassuring throught for the many novices in this new field.

It became clear that the TRS-80 could be used (with our then hypothetical system) to solve problems in many fields where computers were not yet used, mostly because of their high cost.

The IDEA was simple! ALPHA PRODUCT would supply the missing link between the TRS-80 and the "outside world". (more about this "outside world" later)

Early Survival

DANGER! If Radio-Shack entered the same market, we probably would not have survived, but the expectation was that they would be too busy developing their basic line (drives, printers, modem etc.). Thanks to our more specialized products, we would not be competing with them.

BAD START! We began with a failure. Our first product was supposed to be a simple, low cost, general purpose device. It would allow the TRS-80 to accept inputs other than the keyboard. Many kinds of external devices (the "outside world" mentioned before) like photocells, sensors, thermostats, switches, contacts, etc., could be connected easily. In addition, there were two relays to control (on or off) external loads such as motors, lamps, appliances, heaters, etc.. etc.. In other words, it would allow the computer to interact or interface with external devices. We called it the INTERFACER 2. What a mistake! It sounded too much like "expansion interface". Many enthusiastic TRS-80 users called thinking that our "INTERFACER 2" was a low cost Expansion Interface (at $85 that would have been a real bargain!). We wanted to change the confusing name. That meant reprinting the manual, changing the ad, scrapping the flyers, discarding the silk screened cases. Well, "INTERFACER 2" it would stay.

TROUBLE! We also found that the majority of TRS-80 users were AFRAID of the hardware. They could be very comfortable with fancy programming but thought you had to be a computer specialist or technically inclined to put the INTERFACER 2 to work. In truth, some IMAGINATION and a SCREWDRIVER is all you really need. Anyone able to wire a switch could use this device.

WORSE! There was also the fear of plugging a "foreign device" into the precious computer. This notion has all but disappeared as there are now so many quality products designed for the TRS-80 that plugging in a non Radio-Shack device has become common.

Our ad in Creative Computing (80-Microcomputing did not yet exist) hardly paid for itself. We had a decision to make. Were we wrong or just too early? Our first INTERFACER 2 was sold to someone who wanted to, and succeeded in, controlling his fancy model railroad with his TRS-80. Interesting, but what made us stick with the concept was that some of our INTERFACERS began finding use in applications with fascinating possibilities. Space is lacking to describe them, but the most exciting was the successful use of the system in assisting a handicapped young boy. We were pleased to hear of such a meaningful application.

Today

Three years later, as you can see in our ads, The INTERFACER 2 is alive and well. The price went up a bit, and despite the introduction of the more powerful INTERFACER 80, the sales have been steady.

Then came the least understood product! the ANALOG 80. This $139, nicely designed module is an Analog to Digital converter with 8 input channels. Used with your TRS-80, it provides a powerful "data acquisition system". This jargon simply means that you can monitor, measure and record 8 independant varying voltages. Very few people realized its real power. Such a system would have cost over ten thousand dollars just a few years ago.

The possibilities in scientific and engineering environments are endless. This system could replace chart recorders, digital data recorders, programmable calculators, data analyzers and many other specialized and expensive pieces of equipment. Furthermore, up to 8 ANALOG 80's could be used simultaneously for a total of 64 channels of analog input! They simply plug into the TRS-80 using our "X" series of bus extenders (EXPANDABUS).

The idea was simple. We would supply the missing link between the TRS-80 and the "outside world"..........

Our next product was to be a second generation, Input/Output interface, with more flexibility than the INTERFACER 2. Careful design and refinement yielded the INTERFACER 80, the most powerful real world interface on the market today. It has 8 inputs, each optically-isolated and 8 outputs, each with a relay contact. The INTERFACER 80 is fully compatible with our ANALOG 80, allowing these to be used together in order to create systems that control external devices based on "sensed" input under control of the TRS-80.

A FAILURE! in spite of our extensive advertising, very few are aware of the existence of the powerful ALPHA I/O SYSTEM.

The Facts Are:

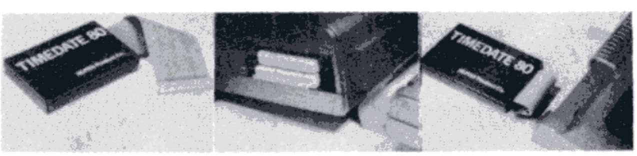
- The ALPHA SYSTEM/TRS-80 combination forms an incredibly versatile and powerful tool for acquisition/processing/control.
- In spite of its moderate cost, the system is sophisticated and reliable.
- The entire system can be easily programmed in BASIC using INP(X) and OUT X,Y commands.
- The modular approach and our EXPANDABUS allow for instant expansion as requirements demand.

The following pages contain more information about the devices mentioned here. We invite you to call or write to discuss your particular application.

Device descriptions; NEXT PAGE

TIMEDATE 80

WHY LOSE PRECIOUS TIME ?



Neat, Compact Design 3 Years Battery Life | Slips Inside E/I (Y Option Shown) | Real Time Without Expansion Interface

- Complete, self contained "true" real time clock/calendar, TIMEDATE 80 continues to keep accurate time and date when the computer is turned off or experiences a power failure.
- **TIMEDATE 80** only needs to be set once, and it's two replaceable "AAA" batteries (not included) keep TIMEDATE 80 running in excess of 3 years. Costly Ni-Cad batteries and charging circuits are eliminated.
- The instant power is applied to the TRS-80, TIMEDATE 80 provides MO/DATE/YR, DAY of WEEK, HR:MIN:SEC and AM/PM information with quartz accuracy.
- **TIMEDATE 80** replaces the computer's internal clock. Extremely useful for automatic operation of remote systems with no operator in attendance. If the power fails and then is restored, only TIMEDATE 80 will update the system with current TIME and DATE information, an impossibility with the computer's internal clock.
- **TIMEDATE 80** is quartz crystal based with INTELLIGENT CALENDAR, including provisions for leap year! TIME display may be by 12 hour AM/PM or by 24 hour military and Eruopean format.
- **TIMEDATE 80** plugs directly into the rear of the TRS-80 keyboard and gives the "TIME$" function even without an Expansion Interface. For those with a disk system, it plugs into the left side panel of the Expansion Interface. An optional "Y" connector can provide for further expansion.
- **TIMEDATE 80's** small size keeps the computer table uncluttered. If you have an Expansion Interface, TIMEDATE 80 literally "DISAPPEARS" by slipping into the empty space in the bottom of the interface.
- Two sets of software, on cassette, come with TIMEDATE 80—"TIMESET" and "TIME$". "TIMESET" is a step by step set of simple instructions for setting TIMEDATE 80. "TIME$" is a set of poke routines which patch DOS and Level II TIME$ to read TIMEDATE 80 and is easily incorporated into any user software. "TIME$" will always print the time and date when LISTING a program—great for keeping track of revisions!
- Other valuable uses for TIMEDATE 80 are: accurate date and time information for business reports like payroll records, financial reports, etc., or to various I/O devices requiring 24 hour clock input, such as laboratory instrumentation, and to communication systems needing "Log In/Log Out" data (bulletin boards).
- **TIMEDATE 80**, fully assembled and tested, 90 day warranty, complete with instructions and software on cassette. $95.00. "Y" option, add $12.00

NEW: Computer to Computer ordering:(212) 441-3755 (24 hr. data line)

CALL
(212) 296-5916
FOR FAST SERVICE

ALPHA Products

85-71 79th Street, Woodhaven, NY 11421

ADD $2.50 PER ORDER FOR SHIPPING AND HANDLING
ALL ORDERS SHIPPED FIRST CLASS MAIL
WE ACCEPT VISA, MASTER CHARGE, CHECKS, M.O.
COD ADD $2.00 EXTRA
QUANTITY DISCOUNTS AVAILABLE
N.Y. RESIDENTS ADD SALES TAX

✓210

typed in. Useful when examining damaged or tricky sectors, the CRC may be regarded as a two-byte checksum. The disk controller starts calculating a CRC whenever an address mark of any kind is encountered during a track write, and whenever a sector is being written or read. On a write, these go to the disk and on a read they are compared to the CRC bytes on the disk, then reported back to the CPU along with an indication of their correctness.

H gives a choice of hex or ASCII dump to the printer starting at any memory address.

According to the manual, possible future additons to Trakcess are a disk compare and a disk search facility. Improvements are promised to keep up with the state of the art in protected disks. An upgrade to the latest version may be had at no charge by sending a diskette with a postpaid mailer to The Alternate Source. Presumably, you would also have to furnish proof of purchase.

Mr. Baker says he doesn't like protected software because it makes modifying or moving the code difficult, and it is usually a pain to back up and use. Baker maintains that, with 300,000 or more TRS-80's in the field, and the fact that relatively few owners are in close contact with large numbers of other users, and that most owners are willing to spend some money on software, there will be plenty of sales potential even after all the swapping.

Whether or not you agree with Mr. Baker's philosophy, it should be pointed out that reproducing copyrighted software for other than your own use is illegal.

Trakcess is a good program, it works and, though not a replacement for Superzap or similar programs, it can be used for purposes other than copying. Besides that, it will give you a mini-education on the working of the disk controller and the various disk formats possible. ■

Hellfire Warrior
Automated Simulations
Mountain View, CA
$39.95 disk, 32K TRSDOS
$39.95 cassette, Level II, 16K

by Debra Marshall
***80 Microcomputing* staff**

Hellfire Warrior is the second in the Dunjonquest series of full-length fantasy role-playing games produced by Automated Simulations (the first is the Temple of Apshai).

The game is structured much like Datestones of Ryn, the minigame introduction to the Dunjonquest series. Like Datestones, commands are issued by pushing various single keys. Graphics on the '80 consist of a single-dimensional top-view representation of corridors and rooms found in the cave you venture into in your hero's role. The character appears on the screen as a triangle created from graphics pixels, treasures are rectangles, and critters and nasties apppear as squares of different sizes, which disintegrate into shooting rays when they are defeated.

Hellfire Warrior consists of four levels of dungeon. Each level proves increasingly harder to stay alive in. However, experience points are gained with each nastie killed, and your character becomes stronger, more resilient and harder to defeat with each battle.

This game is perhaps unique in the fantasy game field because you have the option of creating your own character or bringing a favorite character from any other role-playing game, computer-based or otherwise, with you to Hellfire to be your hero. The Gamemaster will randomly select a hero's attributes for you if you do not choose to create your own character. Either way, you must name your own character, and I quickly discovered the name must have a hero's ring to it, or the program has some distinctively reprimanding thoughts to share on the subject.

You may increase your character's native attributes (and chances to stay alive) by visiting and spending your money at the Armory, the Apothecary Shop and the Magic Shop. Each shopkeeper has many things of various import for sale, and you should choose carefully, because the weapons, armor and magic you provide your hero will greatly affect his chances of remaining alive in the dungeon. Each shopkeeper sets a price for his goods and is open to haggling, within reason. If you haggle too much or offer too low a price, you may find the shopkeeper raising his prices. Weapons may be enchanted, indeed, *should* be enchanted; elixers, nectars, bloods and salves are available in plenty, but without any explanation or guarantees as to their desirability; magic amulets and talismans are also available to the wealthy (read: successful) adventurer.

The object of the game is simple: Wander through the maze, collect all treasure on all levels, buy advanced magical aids, proceed to higher levels, and eventually rescue the warrior maid Brunhilde held enthralled deep in the dungeon. Exit the maze with Brunhilde, treasure and body intact, and you win. Oh yes—pay the cleric and try not to get yourself killed. While resurrection is possible (and even likely), you always seem to lose a lot in the process. Provisions are made for saving a game or character.

Frankly, one of the most enjoyable things about this game is the Book of Lore. It is nicely illustrated, slickly presented and makes for enjoyable reading, which is saying a lot for something that is software documentation. The directions and explanations are thorough and explicit; you will be well-advised to read the book completely before attempting to play the game. There are room and treasure descriptions, and some secret messages that come in handy. In addition, there is a short scene-setting story that proved to me the people who wrote the documentation are game players and fantasy lovers.

The game proceeds in much the same way as Datestones: You manuever down halls and into rooms searching for treasure and secret doors. At any moment, you may be attacked by a nastie or beastie and have to defend yourself. There are pits and traps, and any treasure box may be boobytrapped.

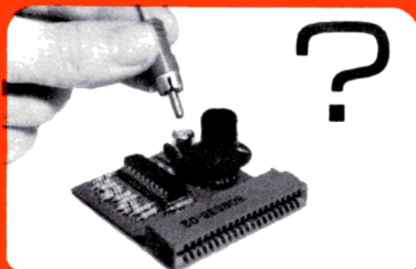
The game is written in Basic and unfortunately, this makes for slow graphics drawing and reaction time; you may continue fighting a battle after you have demolished your opponent because of the slowness of program execution. It is also difficult at times to determine exactly who is coming out ahead in a battle. Unlike the Datestones game, however, critters in this contest do not self-destruct without any effort on your part. In addition, the game is the real time, so don't let your attention wander; you may discover you have been killed while you ran out to the kitchen to get a snack.

The game is not easy to win. The beasties change types on each level, different magic is required, and there is enough going on when the graphics aren't being drawn to keep your interest up. Some of the graphics sections were poorly conceptualized; when it takes many seconds to draw a dungeon section, I don't want to have to watch it being done every three or four steps I take, which occasionally happens. On the whole, however, any true game fanatic will find this game intriguing and entertaining. It is also a welcome change from adventure games without graphics. ■

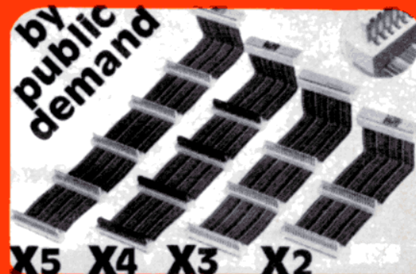
NOW MODEL I AND MODEL III!

Now Model III users can take advantage of the ALPHA I/O system too. Our new MOD III/I BUS CONVERTER allows most port based Model I accessories (such as our ANALOG-80, INTERFACER 2 and INTERFACER-80) to connect to the Model III bus. MOD III/I BUS CONVERTER, complete with all connectors, only $39.95.

STICK-80 MAKES KEYBOARD OBSOLETE.
Features the famous ATARI Joystick. 8 directions + fire control. Simple instructions to make joystick versions of most action games. Plugs into keyboard or expansion int... Price includes ATARI joystick with ALPHA interface and instructions. FREE "MAGIC ARTIST" program..... $39.95
Super Real Time Action Graphic Sound games for Stick 80 by Software Innovations. ALIEN INVASION
COSMIC INTRUDERS. BREAKOUT. Each $9.95
STELLAR ADVENTURE. super action with sound $14.95
Software authors and distributors: contact us for joystick conversion package for your existing games



MUSIC-80 MUSIC-80 MUSIC-80 MUSIC-80 MUSIC-80
Use existing software
or write your own. With this low cost 8 bit digital to analog converter you can synthesize up to 5 music voices. Built-in volume control handy when stereo not near TRS-80. Simply plug the "MUSIC-80" into the keyboard or the E/I screen printer port and connect the output (RCA jack) to any amplifier. The Radio-Shack $12 speaker/amplifier works fine.
Fully assembled and tested. 90 day warranty $39.95

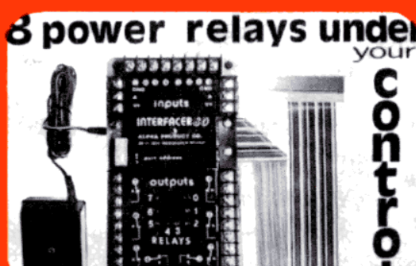


YOU ASKED FOR IT: "EXPANDABUS" X1, X2, X3 AND X4.
CONNECT ALL YOUR TRS-80 DEVICES SIMULTANEOUSLY on the 40 pin TRS-80 bus. Any device that normally plugs into the keyboard edge connector will also plug into the "EXPANDABUS". The "X4" is shown with protective covers (included). The TRS-80 keyboard contains the bus drivers (74LS367) for up to 20 devices, more than you will ever need. Using the E/I, it plugs either between KB and E/I or in the Screen Printer port. Professional quality gold plated contacts. Computer grade 40 conductor ribbon cable.
X2 ..$29 X3 $44 X4 $59 X5 $74
Custom configurations are also available, call us.

ANALOG-80: A WORLD OF NEW APPLICATIONS POSSIBLE.
8 DIGITAL MULTIMETERS PLUGGED INTO YOUR TRS-80!!!
Measure Temperature, Voltage, Current, Light, Pressure, etc. Very easy to use, for example, let's read input channel #4: 10 OUT 0,4 'Selects input #4 and also starts the conversion 20 A = INP(0) 'Puts the result in variable "A". Voila!
Specifications: Input range: 0-5V to 0-500V. Each channel can be set to a different scale.
Resolution: 20mV (on 5V range). Accuracy: 8 bits (.5%). Port Address: jumper selectable. Plugs into keyboard bus or E/I (screen printer port). Assembled and tested. 90 day warranty.
Complete with power supply, connector, manual $139.

INTERFACER 2: LOW COST INPUT/OUTPUT MODULE.
Still the best value in sense/control devices. Use it for energy control, burglar alarm, darkroom, selectric drive, model trains, robots, Skinner box...
—8 latched TTL outputs. 2 relays SPDT 2A. 125V. contacts
—8 TTL/CMOS inputs. Input 0 and 1 are optically isolated.
—Neat and compact design, very easy to use:
10 A = INP(0) 'Reads the 8 inputs (if A = 0: all inputs are low) 20 OUT 0,X 'Controls the outputs and the relays
Assembled & tested, 90 day warranty. Price includes power supply, cable to KB or E/I, superb user's manual, free phone dialer program: $95. Manual only: $5.

LET THE "CHAIN BREAKER" FREE YOUR MINI-DRIVES.
End the daisy-chain mess once and for all. Fits all mini-drives: Percom, Aerocomp, Shugart, Micropolis, MTI, Vista, Pertec, Siemens, BASF. Easy to install: just remove the drive cover, plug in the "CHAIN BREAKER" and replace the cover. Voila!!!
Now you can change and move your drives around without disassembly. Keep the cover on and keep the dust out. High reliability gold plated contacts, computer grade 34 conductor cable. Tested and guaranteed.
Get one for each drive..... only........ $13.95



INTERFACER-80: the most powerful Sense/Control module.
- 8 industrial grade relays, single pole, double throw isolated contacts: 2 Amp. @ 125 Volts. TTL latched outputs are also accessible to drive external solid state relays.
- 8 convenient LEDs constantly display the relay states.
Simple "OUT" commands (in basic) control the 8 relays.
- 8 optically-isolated inputs for easy direct interfacing to external switches, photocells, keypads, sensors, etc. Simple "INP" commands read the status of the 8 inputs.
Selectable port address. Clean, compact enclosed design. Assembled, tested, 90 days warranty. Price includes power supply, cable, connector, superb user's manual $159

GREEN SCREEN WARNING

IBM and all the "biggies" are using green screen monitors. Its advantages are now widely advertised. We feel that every TRS-80 user should enjoy the benefits it provides. But **WARNING**: all Green Screens are not created equal. Here is what we found:
•Several are just a flat piece of standard colored Lucite. The green tint was not made for this purpose and is judged by many to be too dark. Increasing the brightness control will result in a fuzzy display.
•Some are simply a piece of thin plastic film taped onto a cardboard frame. The color is satisfactory but the wobbly film gives it a poor appearance.
•One "optical filter" is in fact plain acrylic sheeting.
•False claim: A few pretend to "reduce glare". In fact, their flat and shiny surfaces (both film and Lucite type) ADD their own reflections to the screen.
•A few laughs: One ad claims to "reduce screen contrast". Sorry gentleman but it's just the opposite. One of the Green Screen's major benefits is to increase the contrast between the text and the background.
•Drawbacks: Most are using adhesive strips to fasten their screen to the monitor. This method makes it awkward to remove for necessary periodical cleaning. All (except ours) are flat. Light pens will not work reliably because of the big gap between the screen and the tube.
Many companies have been manufacturing video filters for years. We are not the first (some think they are), but we have done our homework and we think we manufacture the best Green Screen. Here is why:
•It fits right onto the picture tube like a skin because it is the only **CURVED** screen **MOLDED** exactly to the picture tube curvature. It is Cut precisely to cover the exposed area of the picture tube. The fit is such that the static electricity is sufficient to keep it in place! We also include some invisible reusable tape for a more secure fastening.
•The filter material that we use is just right, not too dark nor too light. The result is a really eye pleasing display.
We are so sure that you will never take your Green screen off that we offer an unconditional money-back guaranty: try our Green Screen for 14 days. If for any reason you are not delighted with it, return it for a prompt refund.
A last word: We think that companies, like ours, who are selling mainly by mail should •list their street address•have a phone number (for questions and orders)•accept CODs, not every one likes to send checks to a PO box•offer the convenience of charging their purchase to major credit cards.
How come we are the only green screen people doing it?
Order your **ALPHA GREEN SCREEN** today $12.50

ALPHA Product Co.

✓210

ADD $2.50 PER ORDER FOR SHIPPING AND HANDLING
ALL ORDERS SHIPPED FIRST CLASS MAIL.
WE ACCEPT VISA, MASTER CHARGE, CHECKS, M.O.
COD: ADD $2.00 EXTRA.
QUANTITY DISCOUNTS AVAILABLE.
N.Y. RESIDENTS ADD SALES TAX.

85-71 79 St. WOODHAVEN N.Y. 11421 ☎ Info and order: (212) 296•5916

Lisp Interpreter
Supersoft Associates
Champaign,IL
$75 cassette
$100 disk

by Gary McGath

An unusual entry in the language field is Supersoft's Lisp interpreter. Lisp is not as well known to microcomputer users as it might be, although it is the most heavily used language in artificial intelligence research. Its simple (though unusual) syntax and powerful data manipulation capabilities make it an excellent choice for dealing with the complicated data structures required by artificial intelligence programs used for natural language processing and pattern recognition.

Other areas where Lisp is effective are symbolic mathematics and adventure-type games. The Supersoft interpreter runs on both the Model I (Level II) and the Model III with 16K of memory; the disk version runs under TRSDOS. All these versions offer TRS-80 owners the opportunity to experiment with artificial intelligence-type programs on their microcomputers.

Lisp stands for List Processing. The only data structure Lisp uses is the list, which is a series of linked data cells. The elements of lists may be other list; this permits the representation of any tree data structure. Programs are lists and may create other lists; memory space for lists is allocated by the interpreter as needed. As a result, it is possible for one Lisp program to create or modify another one.

Supersoft's implementation of the language is fairly complete. Floating-point numbers are used, and a variety of mathematical functions are provided. Variable names (atoms) may be of any length. Property lists, allowing the program to associate attributes with atoms, are supported, as are two different kinds of user-defined functions. Functions that are part of standard Lisp, but can be composed easily from other functions, have been omitted to save space. The user's manual gives Lisp definitions for these functions which can be typed in. The set and reset functions permit screen graphics.

Input/output operations on cassette and disk consist of saving and loading all data are not formally distinguishable.) On the Model III, either 500 or 1,500 baud may be selected for cassette input/output but the rate cannot be changed once it has been set. This limits the Model III user to 500 baud if he or she wants to use the Edit and Trace programs Supersoft provides of tape.

The interpreter is quite fast and nearly bug-free. Garbage collection—the reclaiming of released storage—is the most difficult problem in implementing Lisp; but in this case, the hesitation caused by the garbage collector is barely noticeable, even on a 48K machine.

I have found only two bugs so far in the cassette version. One is typing an atom at the interpreter will not produce a response unless the atom is followed by a space or a right arrow. The other is if an atom is given a property of NIL, that property can never be changed except by directly modifying the property list.

There are, however, a couple of disappointments in store for the buyer of Supersoft's Lisp. The manual is a thin document, typewritten and cheaply bound. It will be an insufficient guide for anyone, even a professional programmer who is less than fluent in Lisp. It does, however, tell the experienced Lisp programmer everything he needs to know.

The major deficiency in the software itself is the lack of any printer output. The only way a printout can be obtained is to save a program on cassette, return to Basic to redirect screen output to the printer, reload the program, and obtain the listing. If Supersoft had provided a POKE function like Basic's, this redirection could have been performed without leaving Lisp.

These problems, however, will not stand in the way of anyone who believes Lisp is the right language for his or her projects. I hope Supersoft will add some improvements in the future, but their product is already an exciting alternative to programming in Basic.■

Ed. note: The AI references are our comments, not the author's.

Spooler
Mumford Micro Systems
Summerland, CA
$16.95

by William C. Huffman

Spooler by Mumford Micro System is a print formatting system for parallel or serial printers. Spooler allows you to specify the number of lines per page, the number of characters per line and a pause, if needed, between pages of output. The pause allows insertion of another sheet of paper if your printer prints only one page at a time. Spooler also allows you to send the contents of the video screen to the printer. This last feature is similar to the JKL function of NEWDOS and others.

Works Without Disk

Spooler is a machine-language program which works on any Model I, Level II, 16K or larger and Model III 16K or larger TRS-80. Spooler will work without disk. Let me repeat that, Spooler works without disk. It is supplied on a cassette containing six copies of the program. Included on the tape are two copies each of the 16K, 32K, and 48K versions of the program. Directions for placing the program on disk are included in the easily read and well documented instruction booklet.

Spooler works by setting aside a user-defined chunk of high memory. All output intended for the printer is intercepted and sent to this area. Whenever Spooler is active and LLIST or LPRINT commands are encountered, the cursor disappears from the video monitor for a brief period. The printed output is rerouted to the reserved high-memory area. When the cursor returns to the screen, you are able to continue execution of your program. You may even load and execute another program while the printer goes merrily on its way producing printouts. The printer continues to run while you continue with your program.

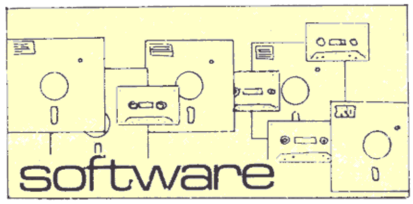
Spooler also works while in DOS. Now you can get a printout of your disk directories if you do not have Newdos or one of the other operating systems with a JKL feature.

I like the indent function of Spooler. I tend to program using very long or multiple statement lines. It is difficult to follow a listing of a program which contains long lines because the line numbers do not stand out. I use Spooler to indent my listings and they become much easier to read.

My Centronics 779 line printer is slow and my Texas Instruments 810 printer will not indent listings. With Spooler, my waiting on a printer is over. In short, I like Spooler. I like Mumford Micro. They sent me an improved version of Spooler several months after I received the original version. The improved version was sent without charge and without my requesting it. I can only give my highest recommendation of Spooler and of Mumford Micro Systems.■

EDTASM-Plus
Mark Chamberlin and Bill Yates
Microsoft
Bellevue, WA
$29.95

by Chris Gundlach



Since the introduction of the TRS-80, those of us wishing to try our hand at Assembly-language programming have had the excellent Radio Shack Editor/Assembler which is well known as a good value because of its "big assembler" features. Unfortunately, using the original EDTASM meant writing programs with the Editor/Assembler, saving the source code on cassette (for later work), saving the object program on cassette, and finally loading the object program and T-Bug (two more cassette operations that cause loss of EDTASM in memory). For me, all the cassette flipping (and the usual hassles with wrong volume settings) took away a lot of creative gusto needed to work with a machine-level program.

Microsoft's new EDTASM-Plus is one of those miracle programs that cures just about every hassle you might think of. If you are just starting to get into Assembly language and the inner workings of a TRS-80, EDTASM-Plus is just what you want.

EDTASM-Plus is written by Mark Chamberlin and Bill Yates; Chamberlin designed the original Editor/Assembler. These people know the TRS-80 inside out, and have been in touch with this computer since it first came out. The knowledge shows, since EDTASM-Plus is an incredibly complete Assembly-language development system oriented to the cassette TRS-80 user, and takes into account the overall design of the TRS-80, the way it uses memory, and so on.

EDTASM-Plus has three important parts and we'll review features of each for you. We'll try to explain things in a way that new Assembly-language programmers can picture. The three parts, all contained within EDTASM-Plus and loaded with one cassette load using System, are: the editor, the assembler, and Z-Bug (a debugging monitor similar in function to Radio Shack's T-Bug).

The Editor

The editor is very much like the same feature in the original Editor/Assembler. Using numbered program lines, it allows you to type in Assembly-language programs in Z80 mnemonics, or code words, such as the command LD A, (HL) which loads the computer's A-register with the contents of a memory location pointed to by the H and L register pair. The editor stores these lines for later conversion, or assembly, into the actual numeric machine-level commands which the Z80 chip in the TRS-80 was designed to work from.

Editing features in EDTASM-Plus have been greatly expanded over those in the original Editor/Assembler. Using a line-editing feature like the Edit function of Level II, you can change program lines. However, EDTASM-Plus also allows you to move any part of your Assembly-language program to another set of program lines. This saves you the headache of retyping one or several lines of assembler code after discovering you had them in a bad place in your program.

EDTASM-Plus' Editor also allows you to edit a given series of program lines without having to reenter the Edit function for every line. You just give the Editor the line-number series and it stays in the Edit mode for each of the lines. Rather than giving the exact range of line numbers, EDTASM-Plus also lets you give a starting line number and a number of lines to edit—a handy feature to use after you've inserted new lines that might be numbered oddly.

Using a variation of this feature, you can also go through your program to add comments to each line. Called the extend function, it lets you move to each line, positions the cursor at the end of each line, ready to input your comment for the line. After you Enter the comment, the Editor moves to the end of the next line—you don't have to retype the Edit command and the X subcommand for each line.

EDTASM-Plus has the other program editing features of the original Editor/Assembler. For example, insert lines starting at a certain number with a certain increment, replace a line, renumber all the lines or a part of them, delete lines, find a string within the assembly code, and so on. Together with the new line, block copying and moving features of EDTASM-Plus, the new Editor is a powerful and versatile working tool for entering Z80 mnemonic code.

EDTASM-Plus also adds a couple of important features not found in the original Editor/Assembler, and which may be new to those familiar with original version.

It will accept macro definitions. A macro is something akin to a DEF FN statement in Disk Basic, by which you define a function that contains a number of actual program statements. When you call the function by name, the computer performs the steps of that function just as if you had written them out. Similarly, a macro is a block of Assembly-language statements that you might wish to use over and over again, but with different parameters or values within the routine. You define, or describe, a macro by writing out the steps using dummy values. The Editor remembers the steps in the macro, and whenever you call it by name and provide the values—a step involving a single program pseudo-op line—it generates the steps of the macro in your Assembly-language program when it's assembled.

EDTASM-Plus also allows conditional assembly. You may establish conditions that must be met for a given part of the program to become part of your assembled machine-language program, and if the condition is not true, the assembly just skips that part of the program as if it were not there. The example given in the EDTASM-Plus manual is a good one: You have a program you wish to assemble to fit a 4K TRS-80 and an extended version of the program to fit a 16K machine. Conditional assembly lets you pluck out the parts you do not want to include in the smaller program, so that you can create both versions using your master program instead of typing in and assembling two versions yourself.

"EDTASM-Plus is one of those miracle programs . . ."

EDTASM-Plus also recognizes new operators within you source mnemonics. Besides using expressions such as VIDEO +40H, you can use multiplication, division, modulo division, and logical operators within your source code, so that the Assembler will calculate and assemble your program correctly without your having to work out the expressions by hand beforehand. You can also use parentheses in expressions, something that was not allowed in the original Editor/Assembler.

The Assembler

The assembler portion of EDTASM-

Plus is the part which reads your mnemonics and converts them to actual digital instruction for Z80. EDTASM-Plus provides a variety of error messages and warnings when your program is assembled, like the original Editor/Assembler.

But the original E/A left you with just a couple of options upon assembly—record the program onto a cassette to reload and run it, or continue working in the Editor. EDTASM-Plus will assemble your program directly into the TRS-80's memory, using a location you choose or one provided automatically by EDTASM-Plus, so that you don't have to record it on cassette to run and debug the program. Using Z-Bug (described next, and part of EDTASM-Plus), you can immediately run your assembly-language program. EDTASM-Plus gives you plenty of the interactive features that are inherent with a Basic interpreter, and eliminates a tiresome and often discouraging series of cassette loads and reloads just to test a program and get it right. This Microsoft package assembles into memory without destroying the Editor/Assembler itself, so you can go back to your source program in the Editor to make corrections and reassemble it to try again. Then, after the program is the way you want, EDTASM-Plus will assemble the program for loading into any memory location you specify, so that you can make your machine-language cassette and use the program without EDTASM-Plus.

The symbol table is alphabetized and error-codes are appended to listings in the symbol table for things such as undefined or redefined labels. Labels used to identify macros rather than actual source-program steps are also identified.

EDTASM-Plus supports lineprinter output, as does the original E/A, so that you can lineprint your source code with or without line numbers, the symbol table, the assembled listing, and so on. Switches (in-line commands given when you assemble the program) allow you to suspend or allow printouts of macros and other parts of the program.

Z-Bug

Z-Bug, the third part of EDTASM-Plus, is like a T-Bug whose IQ was beefed up by some miracle drug! Z-Bug is a *monitor*, a program that allows you to look into the TRS-80's memory location and change them if needed. You use Z-Bug to run your assembled machine language program, see what it does, make corrections and try again, set breakpoints, and jump back to EDTASM-Plus and the source program.

Z-Bug goes beyond T-Bug in that it will display program steps as mnemonics rather than simple byte-by-byte hex numbers. It's like a line-by-line disassembler and because Z-Bug stays resident with your source program and EDTASM, you can reference locations to be examined with Z-Bug using your own symbols rather than specific hex memory locations.

Z-Bug also has a calculator mode so you can perform hex-to-decimal conversion on-line (without disturbing your programs or entering special conversion subroutines) and even calculate expressions. You can ask Z-Bug, for example, to tell you where VIDEO + CURPOS + 40H is by just typing it like that, followed by =. Z-Bug will output numbers in hex, octal, or decimal, and will input numbers from the keyboard in any base from 2 to 16. You can therefore use Z-Bug to type in a memory location bit-by-bit (10011101, for example) and then see what that binary number is in hex, decimal, or octal.

A Byte mode lets you look at individual memory bytes (like T-Bug), and a Word mode lets you see pairs of bytes—properly decoded from the Z80's reversed storage format which has the low-order byte first. Z-Bug allows you to display memory locations as ASCII (character) output, too.

Z-Bug allows eight breakpoints in your program and you do not have to fix a breakpoint after it's reached, as you do with T-Bug. Z-Bug also allows you to step through your program by single steps which is a lot easier than trying to test a program on the fly at microsecond speeds.

Microsoft has included a stand-alone version of Z-Bug on the back side of the EDTASM-Plus cassette, so you can use Z-Bug alone. Symbolic references, of course, work only with a source program, resident with EDTASM-Plus, so the stand-alone version of Z-Bug can't reference symbols.

Z-Bug is great just for looking at the TRS-80's ROM subroutines and the program routines in the reserved RAM areas, so you can learn something of how Basic does what it does.

The EDTASM-Plus Handbook

The instruction handbook is as excellent as the software itself. It's written by William Barden, the author of Radio Shack's book on TRS-80 Assembly-language programming, thereby making EDTASM-Plus a fine companion to your first explorations in machine language. Some assumption of Z80 programming and the original E/A are assumed in Microsoft's manual, but all of EDTASM-Plus' new features are explained thoroughly and well. Microsoft even included a notice of a couple of obscure bugs that would arise in tricky assemblies, and tells you how to use Z-Bug to make the corrections to EDTASM-Plus and punch yourself a corrected tape. Needless to say, EDTASM-Plus obviously represents a huge amount of work by its programmers and Microsoft. ■

The Southeastern Textan
Southeastern Software
Birmingham, AL
$40, cassette Model I

by Dennis Thurlow

Find a need and fill it! Yep, those six words are the secret to success alright, and Southeastern Software has got a winner here! From the well written, indexed(!), easy-to-follow manual to the flawless execution of each and every command, this has got to be one of the best thought out packages I've seen.

The program locates itself to the top of available memory, then asks if you want to type in, or CLOAD, a program. Either way, when you're ready, you have an incredibly powerful editor at your disposal.

Using the shift key and down arrow as a control key, there are 24 cursor control commands that allow you to insert or delete characters, words, or lines of Basic, search for any character, replace any character with any other, or load another program. The control X command puts you in an extended mode that allows eight more commands including string search and replace, block delete, insert lines with automatic numbering, and displaying the amount of free memory.

In addition to the text handling features there is an abbreviated keyboard. Pressing shift and any letter types a complete Basic command or statement.

The repeat key feature works for any input not related to editor commands, the break key will abort a bad load, and the non-destructive cursor can be moved to any point on the screen using the four arrow keys. Complete error messages are displayed on screen and thoroughly explained in the previously praised manual, and programs can be merged in the buffer. When you exit the editor, all text is fed to a compiler that returns it to Basic compression codes.

This is a dream utility for the Basic programmer. ■

Planet Miners
Avalon Hill
Baltimore MD
16K, Level II, Mode's I & III
$14.95

by Darren DeVigili

Planet Miners is Avalon Hill's first effort at a microcomputer game outside the strategic war genre, and is an excellent example of thorough programming. While Avalon Hill is a well established game company, I was a little leery of their excursion into the world of microcomputer game design.

The game's scenario is a futuristic Gold Rush. You have five ships, three opponents, and a limited amount of mining claims to try for. To add realism to the game, the playing options include claim jumping, protesting claims, and (my personal favorite) sabotage.

The game is for zero to four players; zero being the computer versus itself. (The computer always plays in the capacity of the Space Patrol regardless of the number of human or non-human players). All the play options are interactive, for example, jumping a claim on Pluto causes the computer to evaluate the location of your ships, the Patrol's ships, and the claim holder's ships. It also evaluates the finesse of these ships, each player's political pull, and current popular opinion concerning your family. (Each player's ships are considered to be part of a corporate family.) Those factors concern not only claim jumping, but just about every aspect of the game, and its outcome.

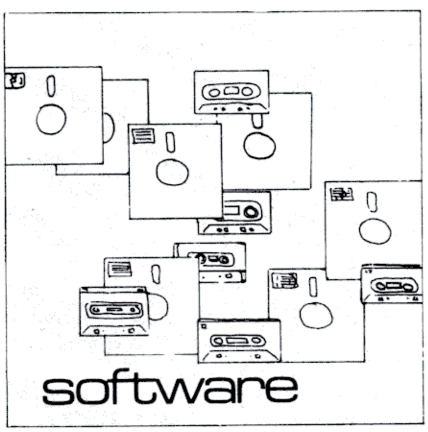
I have noticed two detachments from reality in the game. One is that there are 10 planets in the solar system (the addition being Ceres), and two, once you set a ship's destination, you can't alter it (after take off), until the ship arrives. This adds a degree of spice in some situations.

There are 10 play options available: Large solar system (map); small solar system; display travel times (between planets); display mining status; display ships in orbit; display ship status (yours); set ship destination; protest a claim; attempt to claim jump; attempt sabotage; finished with commands for today (not exactly my idea of an option, more like death and taxes). When attempting sabotage or to claim jump, keep in mind that the Space Patrol may arrest you if you are caught in the act. If there are no Patrol ships around and your opponent catches you, your ship crew will be detained and tortured by your would-be victim. The scars still show, and I've become a wary miner indeed. (I told you it was realistic).

"While Avalon Hill is a well-established game company, I was a little leery of their excursion into the world of microcomputer game design."

If you play with fewer than four players, the computer automatically assigns names to the other players, which are simulated by the computer. This may not seem like much, but I still hate the Lysanders—they win too much.

One game rule worth special mention is that just because someone types their name in first doesn't mean they will play first—the sequence of turns is randomly chosen every round. This is the only program I've seen with this feature. Planet Miners is an exceptional buy.■



Disk Storage: It's No Place to Compromise.

A dependable disk system means everything to the high-performance operation of your computer. At Percom, we know this. And we do something about it.

For example, the connectors on our TFD drive systems are gold plated.

Why? Because solder-plated contacts oxidize, forming an insidious insulation that can cause erratic operation. And then total failure.

Gold-plated connectors are just one way we make better drive systems than the competition.

Now #1 for the Model III

Quality Percom TFD drives provide more features, cost less. Add-on drives start at only $439.00. Complete First-Drive Systems start at only $749.95. Features: **"Flippy" Capability** — Record your programs and data on either side of a diskette. **Greater Storage Capacity** — Store 180 Kbytes — formatted — on one side of a 40-track TFD drive. Store over twice as much on a TFD 80-track drive. **Think Megabytes** — Your TFD drive controller will handle up to four drives. You can access almost 1.5 million bytes of on-line program and data files. **Either Add-in or Add-On** — Your TFD First Drive System can be **either** internal (add-in) or external (add-on). In either case, you get a complete system including the disk controller, drive, power supplies and disk-operating software. **BASIC-Language DOS** — Included on diskette with each First Drive System is Percom's OS-80/III™. This easy-to-use disk-operating system loads into 7 Kbytes of RAM and then frees the drive for other use. With OS-80™ programs, you can have full, read-write interchangeability between Model I and Model III diskettes. **Or Model III TRSDOS*** — TFD 40-track drives work with Tandy's Model III TRSDOS, without modification. For 80-track operation, TRSDOS can be easily modified with diskette patches supplied.

Still #1 for the Model I

Besides greater storage capacities, more quality control measures and lower prices, all Percom Model I drives are rated for double-density operation.

Plug a DOUBLER™ in your Expansion Interface and enjoy the same double-density disk storge capacity as Model III owners. Included with each DOUBLER is a TRSDOS compatible double-density disk-operating system. DOUBLEZAP programs are available for upgrading other popular DOSs for DOUBLER operation. And our double-density version of OS-80™ costs just $49.95.

Of course you don't **have** to upgrade your Model I for double-density operation to use Percom disk drives. But it's nice to know you can.

Percom TFD drives for the TRS-80* Model I are available in 40-, 77- and 80-track versions, in 1-, 2- and 3-drive configurations. Prices start at $399.00.

System requirements: Model III TFD drives work with a 16-Kbyte system (min) and Model III BASIC. The initial drive must be a first-drive system. An optional interconnecting cable is available for expanding with external drives #3 and #4. Model I TFD drives work with a 16-Kbyte system (min) equipped with an Expansion Interface, Level II BASIC and DOS software, and an interconnecting cable. Two- and four-drive interconnecting cables are available from Percom.

To order products, or for the name of your authorized Percom retailer, call toll free: **1-800-527-1592.**

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

Yes . . . I'd like to know more about the best for my TRS-80 computer. Send me free literature about quality Percom products.

☐ Model I ☐ Model III

Send to
PERCOM DATA COMPANY, Inc., Dept. 8
211 N. Kirby St. Garland TX 75042

name ____________________

street ____________________

city ____________ state ________

zip ____________ phone number ________

MAIL TODAY!

M01

™trademark of Percom Data Company, Inc.
*trademark of Tandy Radio Shack Corporation which has no relationship to Percom Data Company.

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

TRS-80 Assembly Language
Herbert S. Howe, Jr.
Prentice-Hall, Inc.
Englewood Cliffs, NJ
$15.95 Hardcover, 186 pp.
$6.95 Softcover

by Edward D. Young III

Learning how to write TRS-80 Assembly-language programs can be a very difficult task. Unlike Basic, in which the instructions bear a logical relationship to the operation intended, Assembly-language instructions usually resemble the names of exercises rather than computer operations. For example, you can Push or Pop a byte, Rotate or Shift a bit, etc. Hubert Howe, Jr., a columnist for the *TRS-80 Monthly News Magazine,* offers his book as a guide to understanding this complex, yet elegant, language. Except for the few reservations noted below, this book is a good reference for both beginning and experienced Assembly-language programmers.

Howe divides his book into two sections. Part I introduces and explains the basic concepts of Assembly-language programming. In addition to describing the operation of the Z80, the heart of the TRS-80, it explains the organization of the TRS-80's memory and how to use the various permanent subroutines which reside in it.

The discussion of how certain input/output devices, such as the keyboard and video display, are linked to special memory locations is particularly informative. Other chapters in Part I introduce the Z80 instruction set, the Z80 stack, and Radio Shack's Editor/Assembler program.

Howe assumes that the reader has access to a 16K Level II and has Radio Shack's Editor/Assembler or an equivalent assembler (such as the Apparat EDTASM which comes with NEWDOS +).

An assembler allows you to assemble and execute programs. If you do not have one you may still be able to load and execute the programs in this book by using POKE or by using a machine language monitor program such as T-Bug. In any event, the book will be clearer if you can assemble the sample programs and execute them on your own computer. This fact notwithstanding, the chapters in this section are clear, instructive, and a joy to read.

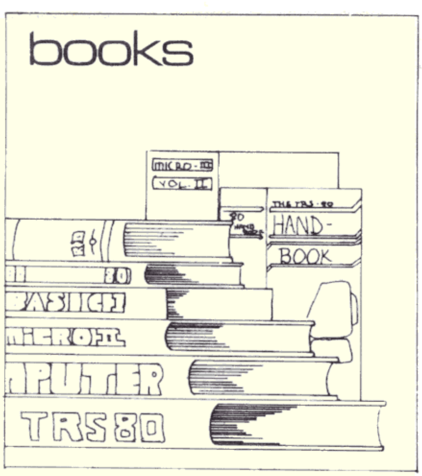
Illustrating Practical Tasks

Part II of the book illustrates the practical tasks that can be performed using Assembly-language programs. There are chapters on how to input and output data efficiently, read a cassette tape written in any format, move large blocks of data, and perform floating-point and integer arithmetic. Three chapters, however, are particularly outstanding. Chapter 15 describes, in great detail, how to use Assembly-language subroutines in conjunction with Basic programs. This chapter culminates in the presentation of a subroutine which allows the user to sort 100 alphanumeric strings in less than two seconds!

Chapters 16 and 17 discuss disk input/output and disk files. Chapter 16 contains a concise explanation, in simple terms, of how a disk is formatted and accessed. It also points out errors in the documentation of TRSDOS. Chapter 17 reveals how to store and retrieve information from seven different types of disk files. This chapter alone is worth the price of the book. Using the information contained in this chapter you can read virtually any disk, even one protected by a password.

On the negative side, a glossary and index are notably absent from the book. Both of these items are important in an introductory guide, since beginners cannot be expected to retain everything they read the first time through. A reader trying to find the definition of "two's complement", for example, must thumb through the book before discovering that the definition is on page 85. Also, there are a number of typographical errors in the book—a minor distraction. Finally, this book is not typeset in the usual manner. Rather than using the easy-to-read typeface found in most books, the author proudly announces that this book was composed and printed using a Diablo Hy-Type I printer with a Model I. The author probably thought that it was a good illustrative example to use a computer to write a book about computers; I found the print hard to read.

Nevertheless, I think this book will prove valuable to TRS-80 owners. It is written specifically for the TRS-80; it takes advantage of the short-cuts and features designed into the '80; and it provides helpful hints of practical importance to programmers. ■



The Nature of Computation: An Introduction to Computer Science
Ira Pohl and Alan Shaws
Computer Science Press
Rockville, MD
Hardcover, 386 pp.
$16.95

by Joel Benjamin

The Nature of Computation: An Introduction to Computer Science is a rigorous introductory textbook to computer science and is for anyone just beginning in the field.

The authors have used preliminary drafts of this book as the text in college classes over the past ten years. They state their objective is to provide the following:

1. A survey of the field;
2. Initial literacy of the language and methods found in computer science, and;
3. A historical, philosophical, and social perspective.

The authors have succeeded admirably in achieving these goals.

Algolic

The book has been organized logically and coherently. First, we are presented with a taste of what programming is like with the use of a machine-independent, high-level language known as Algolic. It is similar to Pascal and is particularly suited to expressing and executing algorithms. A number of well-known algorithms are examined, and we are shown how to express them in Algolic.

An Account of Techniques

We are then treated to an interesting

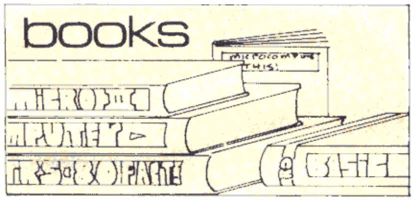
Reader Service for facing page ✓97→


StarFighter
The PENULTIMATE Space War Game is HERE!
TRS-80*
Model 1 & 3
16K and up — Tape
010-0102 . . . . $24.95
32K and up — Disk
012-0102 . . . . $29.95
ai Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.
BOX 3435 • LONGWOOD, FL 32750 • (305) 862-6917
ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172
SHIPPING & HANDLING CHARGES ARE EXTRA
PRICES SUBJECT TO CHANGE WITHOUT NOTICE
*TRS-80 is registered Trademark of the Tandy Corp
PHOTO COPYRIGHT - DON DIXON

account of the development of computational techniques from the abacus to the intricacies of the structure of a modern computer. Along the way, number bases, floating point numbers, strings, arrays, Boolean algebra, hardware components and circuits, digital logic, and computer architecture are all presented and explained in a remarkably lucid fashion.

Programming languages and operating systems are then explored. In this section we get an insight into the relationship between high-level, Assembly, and machine languages; the use of compilers, interpreters and assemblers, and the various examples of different languages suited to particular programming needs.

Next, the authors introduce us to the theory of Turing Machines, computability and algorithms.



The book ends very appropriately with a discussion of the social and ethical questions raised by the various ways in which computers are used in our society.

There is a set of comprehensive excercises and questions at the end of each chapter as well as suggestions for additional readings. These aids, along with the comprehensive nature of the text, make *The Nature of Computation* ideal as a textbook in an introductory college course. But it can also be used by an individual who wants a painless and clear introduction to computer science, which can be read on his or her own. ■

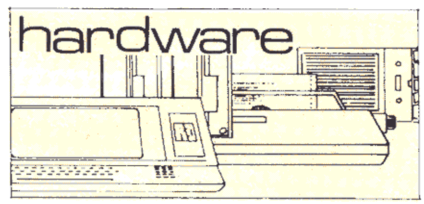
DIP-81
DIP, Inc.
Boston, MA
$499

by David Tinis

Aside from the computer itself, the single largest purchase a hobbyist makes is a printer. Just one year ago a usable impact printer retailed for at least $1,000. But that is changing. By flipping through the ads in any of the magazines it's evident that a printer price war is raging. With that in mind, I went out to find a low-cost, reliable, well built printer. I finally settled on the DIP-81. It seemed to offer the best value for the price.

The DIP-81 is an impact printer capable of printing 100 characters per second bidirectionally on an 80-character line. It can use rolls, fanfold or single sheets up to 8.5 inches wide. Paper feed is by friction roller only. The normal print mode is a 7 by 7-dot matrix but can be expanded under software control to a 14 by 7 matrix.

The DIP-81 is not a marvel of advanced technology. It uses proven parts to accomplish a simple task—print. The unit is built on a steel chassis to which all subassemblies attach. Covering it is a single-piece shell of durable plastic. The funny bulge on the right covers a cooling fan that directs air over the electronics and the print head. The print mechanism is manufactured by the Two Day Corporation and uses a single ac synchronous motor to drive both the print head and paper feed mechanism. The print head itself is a seven-wire design that has a rated life of 100 million characters. I was already familiar with this print head when I started looking at the DIP-81. The rest of the mechanism is rated for 10 million lines of printing.



NORMAL 80 CHARACTER 10 PITCH

```
 !"#$%&'()*+,-./01
23456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTU
VWXYZ[\]↑←°abcdefg
hijklmnopqrstuvwxy
```

EXPANDED 80 CHARACTER 5 PITCH

```
 !"#$%&'()*+,-./01
23456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTU
VWXYZ[\]↑←°abcdefg
hijklmnopqrstuvwxy
```

NORMAL 96 CHARACTER 12 PITCH

```
 !"#$%&'()*+,-./01
23456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTU
VWXYZ[\]↑←°abcdefg
hijklmnopqrstuvwxy
```

EXPANDED 96 CHARACTER 6 PITCH

```
 !"#$%&'()*+,-./01
23456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTU
VWXYZ[\]↑←°abcdefg
hijklmnopqrstuvwxy
```

NORMAL 132 CHARACTER 16.5 PITCH

```
 !"#$%&'()*+,-./01
23456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTU
VWXYZ[\]↑←°abcdefg
hijklmnopqrstuvwxy
```

EXPANDED 132 CHARACTER 8.25 PITCH

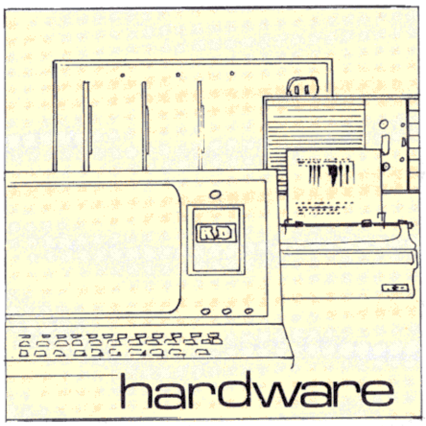
```
 !"#$%&'()*+,-./01
23456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTU
VWXYZ[\]↑←°abcdefg
hijklmnopqrstuvwxy
```

DIP Print Samples

Electronically the DIP-81 is just as simple. All components except the power transformer, bridge rectifier and cooling fan are mounted on a single P.C. board. Overseeing operation is an 8035 microprocessor with the control program in a pair of 2708 EPROMs. An 8155 RAM-I/O controller round out the large scale ICs. Toss in a half dozen TTL parts, some voltage regulators and the TIP122 hammer drivers and you have a simple but efficient design. A quick note about the TIP122s: They were chosen because of their ability to drive inductive loads without the need for clamping diodes to suppress the inductive kickback of the print hammers, a sign of good design practice.

A parallel Centronics interface is standard with the unit though a serial RS-232 interface is available on the P.C. board as an option. Interestingly enough, simple conversion instructions are included in the operation and maintenance manual, but conversion by a non-authorized person voids the 90-day warranty.

There are three operator controls and two indicators on the front panel plus the on/off switch on the back panel. From left to right the controls are TOF (top of form), SEL/DESEL (select/deselect) and Line Feed. Since the DIP-81 doesn't have a separate paper feed motor, pressing TOF initiates a series of line feeds. At power-on, the printer assumes the paper is already at top of form. It then counts lines and if a TOF command is encountered, it advances to the next logical top of form and resets the line count. Line Feed initiates a single print cycle during which nothing is printed, although the print head travels across the paper. Again this is a result of using a single motor for all mechanical functions. It also explains why the DIP-81 is a bi-directional printer. You never know on which side of the printer the print head will start or stop. Having a synchronous motor instead of a more position-precise stepper motor also causes a slight column misalignment during printing. SEL/DESEL determines if the printer is ready or not, if it is, the red select indicator directly above the SEL/DESEL switch will be on. In the deselected mode you can use the DIP-81's self-test feature. Pressing Line Feed and TOF simultaneously causes the printer to display the 96 ASCII characters it is capable of printing. The final indicator is power-on located directly above Line Feed.



I ordered my DIP-81 with the parallel interface and connecting it to my LNW Expansion Interface was no problem. I did cut line 33, printer chassis ground, to avoid its being connected to signal ground in the expansion interface. The unit powered up and operated properly the first time I tried it. The DIP-81 lists for $499 but, like all current printers, is being discounted. I got mine for $395 and have seen it advertised since at $379. In retrospect I am quite satisfied with my DIP-81. While I wouldn't recommend it for highest quality business applications, I do feel it makes an excellent printer for the personal computer owner.■

Home Computer Work Table
Computer Roomers, Inc.
Dallas, TX
$179.95

by Dan Keen and Dave Dischert

When microcomputing was "just a hobby", we were content to stick the computer in a back room of the house. But then we became computer-holics, never leaving the machine except to eat, and go to work. So, we centrally located our TRS-80 in the living room.

Now we needed to solve the problem of what to set it on. We have a Daisy Wheel II which develops a lot of torque and requires a table of steady support, preferably one made out of cast iron or concrete. Daisy is also very wide, which makes placement difficult in some areas of the room.

Since the unit was to be placed in a very visible part of the house, it needed to be decorative.

So the criteria for our table was that it had to be firm, well built, be able to provide a home for peripherals, and provide a work space. It had to look nice and most of all, since we are cheapskates, it had to be reasonably priced.

This describes a new computer table from Computer Roomers, Inc. which has a nice walnut wood grain look with black trim.

The unit came in a box and though it was only about three inches deep, it was wide and heavy! (Plan on inviting a friend over the day you expect delivery.) It was packed extremely well, using big pieces of custom cut styrofoam.

We found that the shelf which is at the rear of the table and runs along its entire width is the exact depth of our Shack disk drive as well as a friend's Apple drive, and we would assume most 5-inch disk drives. This shelf (called a video shelf) is a real plus since it gives you more room on which to place items and yet doesn't take up any more space in the room. The video monitor fits well on the shelf also, along with four drives and cassette recorder (your monitor can't sit on that interface any longer). Personally, we found this preferable to desks which have a shelf underneath to house only the drives. Frequently we have more than one person sitting at the console, and this table has plenty of leg room.

"We . . . enjoy our new table."

An open space along the rear allows for cables and wires to be routed neatly out the back.

It's so easy to put together "even a child can do it" . . . well almost. It took about 20 minutes to assemble. They say "no tools needed," but we did need a hammer to tap a few sectors into their locked position. A wide blade screwdriver is required for disassembly.

While the instruction manual pictures the desk with two "wing" extension shelves, it is apparent that only one at a time can be used. We assembled two tables, and both were missing the necessary "shoulder screws" that mount the second side. So, although a left and right wing come with the unit, plan on only using one or the other.

We thoroughly enjoy our new table, but some computerists may find its lack of drawers a slight disadvantage.■

UCSD Pascal Compiler
FMG Corporation
Fort Worth, TX
$250

by Dennis Thurlow

First, will all of you who program 370's for a living, and are only reading this to find out about "little" computers, please wait on the other side of the room?

Now, I did that because those guys would laugh if I called Basic a language because, in fact, it isn't! Basic is a code that must be interpreted by a large machine-language program called (appropriately enough) an interpreter. Usually an editor of some kind is included to help you build the code, make changes if necessary, and save the code to a storage medium.

A *real* language, on the other hand, is used by a compiler to generate machine code that is directly saved to disk or tape, or automatically executed. Since the system was never meant to be interactive, no editor is included and the program is usually punched onto cards and fed into the computer.

Let me also point out that there are exceptions to the rule. There are Basics that compile, and versions of languages, such as Cobol, that run on interpreters. And then there is Pascal.

The Language

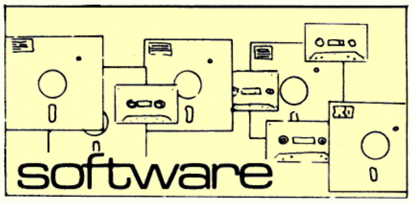
Pascal incorporates all the standard programming features found in most languages, but the syntax is much easier to learn. Example 1 presents a typical Pascal program and Example 2 is the same program in a non-structured format. Both run identically, compile to the same number of bytes, and follow the same syntax rules, as laid out in the syntax tables in Examples 3–5.

Example 3 indicates that a program must start with the word Program. Any time a word or character appears in a balloon (rounded corners) it must be entered literally. You can see that the word Program is in both Examples 1 and 2.

Next an identifier is required. The square corners indicate that options are available and that there is probably another syntax table to define those options. In this case Example 3a is the syntax table for identifiers. In Example 1 the identifier is the word Test.

The identifier in parentheses usually defines whether I/O is going to be used. Note that all statements in Pascal end with a semicolon. The program itself ends with a period.

Example 4 is the syntax table for a block. It provides for variable definitions and the procedures used in Examples 1 and 2. To define a statement you must use the syntax table in Example 5. You can

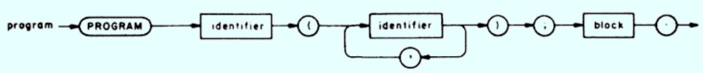


also define procedures and functions that will be used repeatedly.

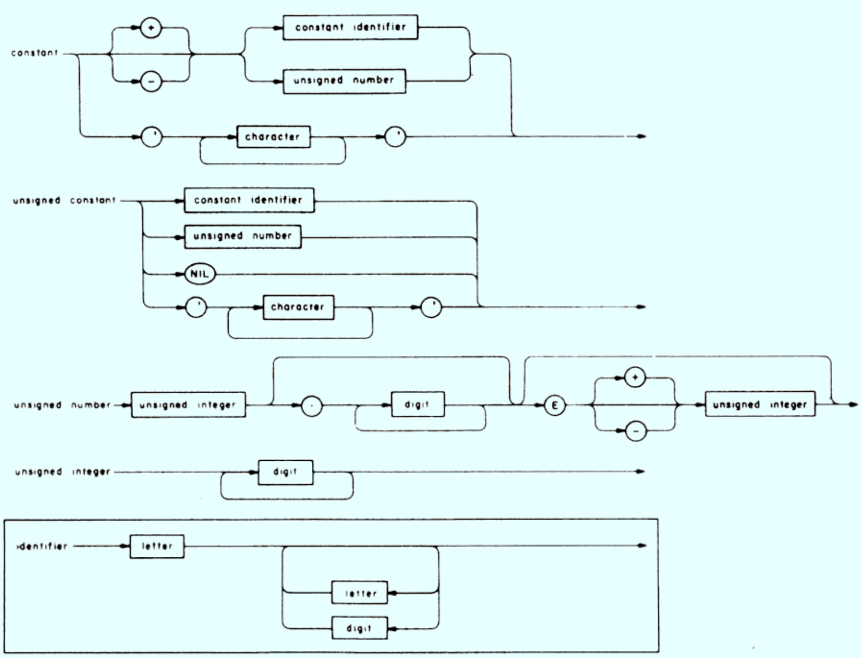
Structuring is important when you develop large programs that someone else may have to maintain, or that take so long to write you forget what was going on in the different sections. Professional programmers, in particular, tend to favor structured programming, particularly

```
PROGRAM TEST (OUTPUT);BEGIN VAR X: INTEGER;REPEAT WRITE(X,
SQUARED = ',SQR(X));WRITELN('THE SQUARE ROOT OF',X,'IS',
SQRT(X));UNTIL X>100;WRITELN('DONE')END.
```

Example 2.



Example 3.



Example 3a

```
PROGRAM TEST (OUTPUT);
  BEGIN
    VAR
      X:INTEGER;
    REPEAT
      WRITE(X,'SQUARED = ',SQR(X));
      WRITELN('THE SQUARE ROOT OF',X,',SQRT(X));
    UNTIL X>100;
    WRITELN('DONE')
  END
```

Example 1.

when it is this easy!

The FMG package is on three disks. The modules are arranged to allow for maximum free space on the disk, which is usually about 30K. Interaction with the user is achieved with a command prompt line upon boot up. From this the user can call up:

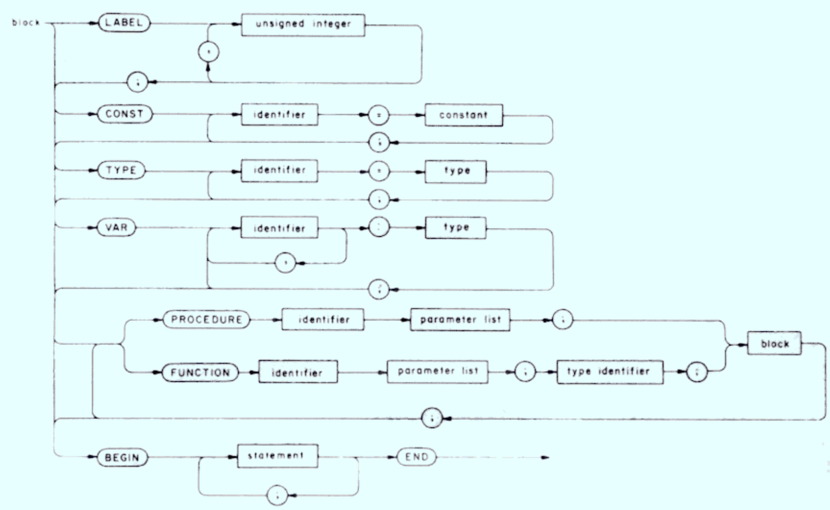
1. L(ink): The linker grafts routines from the System Library to the program that is to be compiled. You may also link to machine-language routines that you have written yourself! The advantage of this is that only those routines that are necessary to run the program are linked. This results in a tremendous saving of space and time.
2. C(omp): The compiler saves the compiled program to disk.
3. X(ecute): Execute a program that is already linked and compiled.
4. R(un): Compiles a program that has been linked, and runs it instead of saving it.
5. F(ile): The filer handles all the disk I/O that a DOS would take care of in othe interactive systems. This is one of th two routines added by UCSD to make Pascal interactive.
6. E(dit): This editor is specifically for constructing Pascal code. By inserting a different disk you could call the editor for the assembler, also supplied. The keyboard is debounced, but is a little slow.

All the modules work very well, and generate nice, compact code, compared to Microsoft's Basic compiler, for instance. The program in Example 1 takes 1407 bytes of code when compiled and linked. This may appear to be quite a bit when compared to the 198 bytes of Pascal code, but remember that most the overhead is constant, and as your programs get longer, the overhead gets less significant. And it's still quite a savings over the 14K the same program takes compiled with Microsoft's Basic compiler.

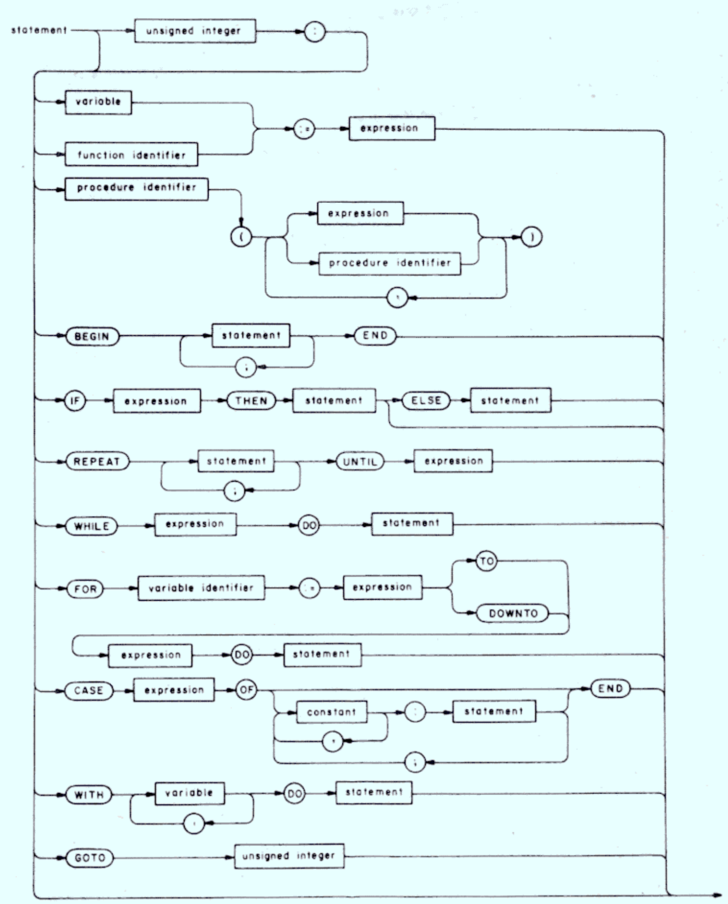
The Documentation

The sore point of the package is the documentation. The manual is a 284-page looseleaf folder. It was written to supplement a teacher's instruction, and does *not* teach you Pascal. You must know Pascal or be learning it from another text (or teacher) to use this compiler.

Another problem is that the manual is the same one used with the PDP-11 and CP/M versions of the package. Most of the examples and tables are useless unless you keep flipping back to the additions in the front of the book. And, lest you get your hopes up, let me say that there are only seven of those pages.■



Example 4.



Example 5.

THE ASSEMBLY LINE

by William Barden, Jr.

"Look on these next two columns as a sort of "Cliff Notes" about the book—a supplement to those six pages."

I'm currently reading *Going Ahead With Extended Color Basic.* I was even amused by the drawings of the Color Computer with legs, the result of inbreeding in the Radio Shack computer line. Everything was fine until I got to page 144. Even though it was an exciting plot, the next six pages took several days to get through. Look upon these next two columns, then, as a sort of "Cliff Notes" about the book—one that you can read to supplement those six pages.

These columns are meant to give Color Computer assembly language users some hope. Model I and III users may wish to read them to see how the 6809 compares to the Z80.

First let me say that the *Extended Color Basic* book is very well written—Jonathan Ericson at Radio Shack deserves credit. The book is geared to the first time user, and it's very difficult to present detailed machine language information in six pages.

We'll start from the ground up, so the first-time Basic programmer may be able to get into machine language programming with a minimum of grief. Experienced programmers, bear with me for a while.

6809 Machine Language

The Color Computer uses the 6809 microprocessor which is an upgrade of the popular 6800. Both microprocessors are manufactured by Motorola. The 6809 instructions include the 6800 instructions as a subset on an *assembly* language level.

This means that both instruction sets would have similar instruction mnemonics, such as CLRA, for Clear A, but that the binary value for instructions might be different between the two. One important implication of this is that you can't pick up the *machine code* for a 6800 program and run it on the Color Computer—it probably won't work. You might be able to pick up the assembly-language code for a 6800 program, however, feed it into the Color Computer assembler, and get resulting *object code* that works.

One Hexadecimal Digit Or four Bits

Nine Bytes Of Machine Code

BD, B 3 , ED,
44,
56,
BD, B4, F4,
39 — Two Hex Digits, 8 Bits, Or One Byte

HEX	*BINARY*	*DECIMAL*	*HEX*	*BINARY*	*DECIMAL*
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	A	1010	10
3	0011	3	B	1011	11
4	0100	4	C	1100	12
5	0101	5	D	1101	13
6	0110	6	E	1110	14
7	0111	7	F	1111	15

Fig. 1. 6809 Machine Code

Machine Code	Assembly-Language Code			
	LABEL COLUMN	OP-CODE COLUMN	OPERAND COLUMN	COMMENTS COLUMN
BD,B3,ED	START	JSR	$B3ED	FIND INTEGER
44,		LSRA		SHIFT A
56,		RORB		ROTATE B
BD,B4,F4,		JSR	$B4F4	FIND FLOATING
39		RTS		RETURN

Fig. 2. Machine vs. Assembly-Language

A sample of 6809 machine language is shown in Fig. 1. This happens to be a short segment of code that will divide the contents of a 16-bit value by two. The code is in hexadecimal, which is a shorthand way of representing binary data. One hexadecimal digit represents four binary digits, or bits. Two hexadecimal digits represents two groups of four binary digits.

The basic number of bits in the Color Computer and 6809 is eight, constituting one byte. All memory and CPU operations generally transfer data and perform operations on eight bits or one byte of data. Eight bits can hold binary values of 00000000 through 11111111, the decimal value of 0 through 255, or the hexadecimal value of $00 through $FF. The prefix $ is used in the 6809 and other microprocessors to indicate that the following data is in hexadecimal. Basic uses &H in lieu of the $.

If you're hazy about binary and hexadecimal, you'll have a tough time following anything from this point. Proceed as follows: Go to a chapter on binary and hexadecimal in any basic computer text and practice some conversions between binary, decimal, and hexadecimal. You don't have to spend hours in practice, but get a nodding familiarity.

Back to the machine language . . . A machine language instruction consists of one, two, three, or four bytes of data that the 6809 will recognize as an instruction. The instructions will range from

such simple operations as putting a zero into a CPU register, on up to a multiply. The instructions from Fig. 1 are shown in Fig. 2 with the operations they represent.

Assembly Language for the 6809

Fig. 2 shows the machine-language codes on the left, and the equivalent assembly on the right. The *mnemonics* are just what they sound like (from Mnemosyne, the Greek goddess of computer writers)—abbreviations for 6809 instructions. For example, it's much easier to write LDX instead of Load the X Register.

The mnemonics for the instructions are called *op codes,* short for operation codes and are in the second column of the assembly language. The third column figure is the *operand* column. Certain instructions require no operands, while others require several operands. To transfer data from a memory location to the X register, for example, an LDX $2000 would have the mnemonic LDX and an operand of $2000.

The first column of the assembly language represents a label field. The location of an instruction might be labeled with a label such as LOOP and subsequent instructions could refer to the location by the label, rather than an absolute address, such as BEQ LOOP, which would be a "Branch on Equal to location LOOP."

The last column holds applicable comments.

The portion of the figure representing the labels, op codes, operands, and comments field in Fig. 2 is the assembly language. The portion on the left with the hexadecimal values is the machine language. The assembly language code, called the *source code* is translated into the proper machine language by an *assembler* program for the Color Computer. (By the time you read this, the Radio Shack Color Computer assembler should be out.) The assembler decodes the symbols into the proper machine language instructions.

Not only is it possible to translate from assembly language into machine language, but it's possible to go the other way, from machine language into assembly language. A *disassembler* program will examine a machine language program and produce the mnemonics and operands. There are several disassemblers available currently for the Color Computer.

Hand Assembling

In the remainder of this discussion we'll be using examples of machine language rather than assembly language. It is possible to bypass an assembly process by doing hand conversion of what looks like assembly language code. An example is shown in Fig. 3. The instructions are first written down with their op codes and operands. Next, each instruction is "roughed out" as far as the number of bytes in the instruction. Next, the op code values are filled in, followed by the values for the operands.

This process is not a simple one. It requires a good book on the 6809 instruction set, and some study about the CPU registers, instructions, and addressing modes. The bible for this instruction is not Barden, or Leventhal, or Warren, but Motorola. Motorola's *MC6809 Preliminary Programming Manual* is available from Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721.

Next let's make certain we know something about the internal architecture of the 6809. The 6809 chip has a number of *registers.* A register is nothing more than an eight or 16-bit memory location that is located in the CPU, rather than in RAM memory.

The registers are assigned names that are related to their function. Fig. 4 shows the 6809 registers that are accessible to the programmer. There are other registers, of course, but we'll leave an excruciatingly detailed account of the internal operation of the memory fetch cycle and the associated registers to Adam Osbourne.

6809 Registers

The A and B registers are eight-bit registers that hold the results of arithmetical, logical, and other types of operations. They are called *accumulators,* and are used to process data by adds, subtracts, shifts, exclusive ORs, and so forth. A and B can be lumped together to form one 16-bit register called the D register to allow operations on 16 bits at a time.

The PC is the Program Counter register. It points to the next instruction byte. For a machine language instruction

Step 1: Write Down Instructions

LOCATION	*CONTENTS*	*CODE*
$2000		START STA $2000
		STX $2001
		RTS

Step 2: Find Number Of Bytes

LOCATION	*CONTENTS*	*CODE*
$2000		START STA $2000
$2003		STX $2001
$2006		RTS

Step 3: Fill In Codes

LOCATION	*CONTENTS*	*CODE*
$2000	B7 20 00	START STA $2000
$2003	BF 20 01	STX $2001
$2006	39	RTS

Fig. 3. Hand Assembly Example

16 BITS

8 BITS

X INDEX REGISTER

Y INDEX REGISTER

U USER STACK POINTER

S HARDWARE STK PNTR

PC PROGRAM COUNTER

A B D REGISTER

DP DIRECT PAGE REGISTER

E F H I N Z V C CONDITION CODES

Fig. 4. 6809 Programming Registers

of three bytes, the PC will increment three times as the three machine language bytes are assembled in the CPU. This is the fetch cycle. At the end of the fetch cycle, the PC points to the next memory location, which must hold another instruction.

At the end of the fetch cycle, the assembled instruction is executed. This might involve reading an eight or 16-bit operand from memory and adding it to the A register, loading the B register with the contents of the A register, or many other operations. The fetch and execute cycles are executed automatically with each instruction—for the time being, simply forget about the two separate parts of an instruction. About the only consideration is that more involved instructions take longer to execute; this is sometimes a factor when you're figuring out timing loops or trying to speed up programs.

The S, or Hardware Stack Pointer, points to the stack area of memory. Stack can be located just about anywhere the programmer desires (ROM is a bad choice). Usually 100 bytes of stack is sufficient. The stack (see Fig. 5) is an area used to store return addresses for subroutine calls, temporary data from the CPU registers, or *interrupt addresses*. Branching to a subroutine saves the return address on the stack, for example, and a subsequent RTS, or Return from Subroutine, instruction retrieves the return address from the stack. The stack builds down from high memory to low memory as addresses or data is pushed onto it.

The U register, or User Stack, is a second stack pointer that points to a user-specified stack. The S register is a hardware stack, since it is related to the built-in hardware functions that store data in the area pointed to by S, such as "branching to subroutines." The U stack area may be redefined constantly for the programmer's convenience.

The X and Y registers are *index registers*. They are used to point to the location of memory operands. Operands beyond or prior to the pointer may then be easily accessed by an indexed-type instruction, such as "LDX +20,X", which loads the contents of a location pointed to by the index register *plus* 20 locations. The X and Y registers are continually loaded with new values as new blocks of data are accessed.

The DP, or Direct Page register, is used to define *pages* of 256 bytes. This is an optional addressing mode that allows shorter instructions (using less memory and time). We won't be using this mode in the examples here.

The Condition Code register is an eight-bit register that is really a conglomeration of eight bits. These condition codes are set (1) or reset (0) according to the actions of certain instructions. The Z condition code, for example, is set if the result of an add (and many other instructions) is a zero. The condition codes may be tested by conditional jumps and branches that follow the operations. This is the main way of altering the path of a program. A "BEQ $2000", for example, branches to an instruction at location 2000 hex if the result of the last operation

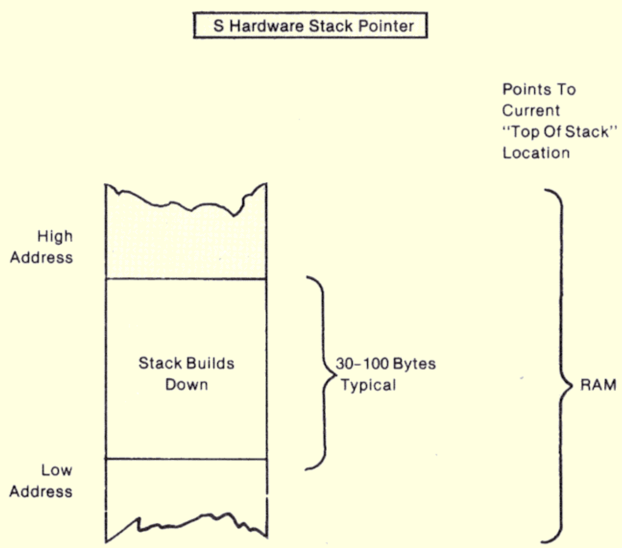
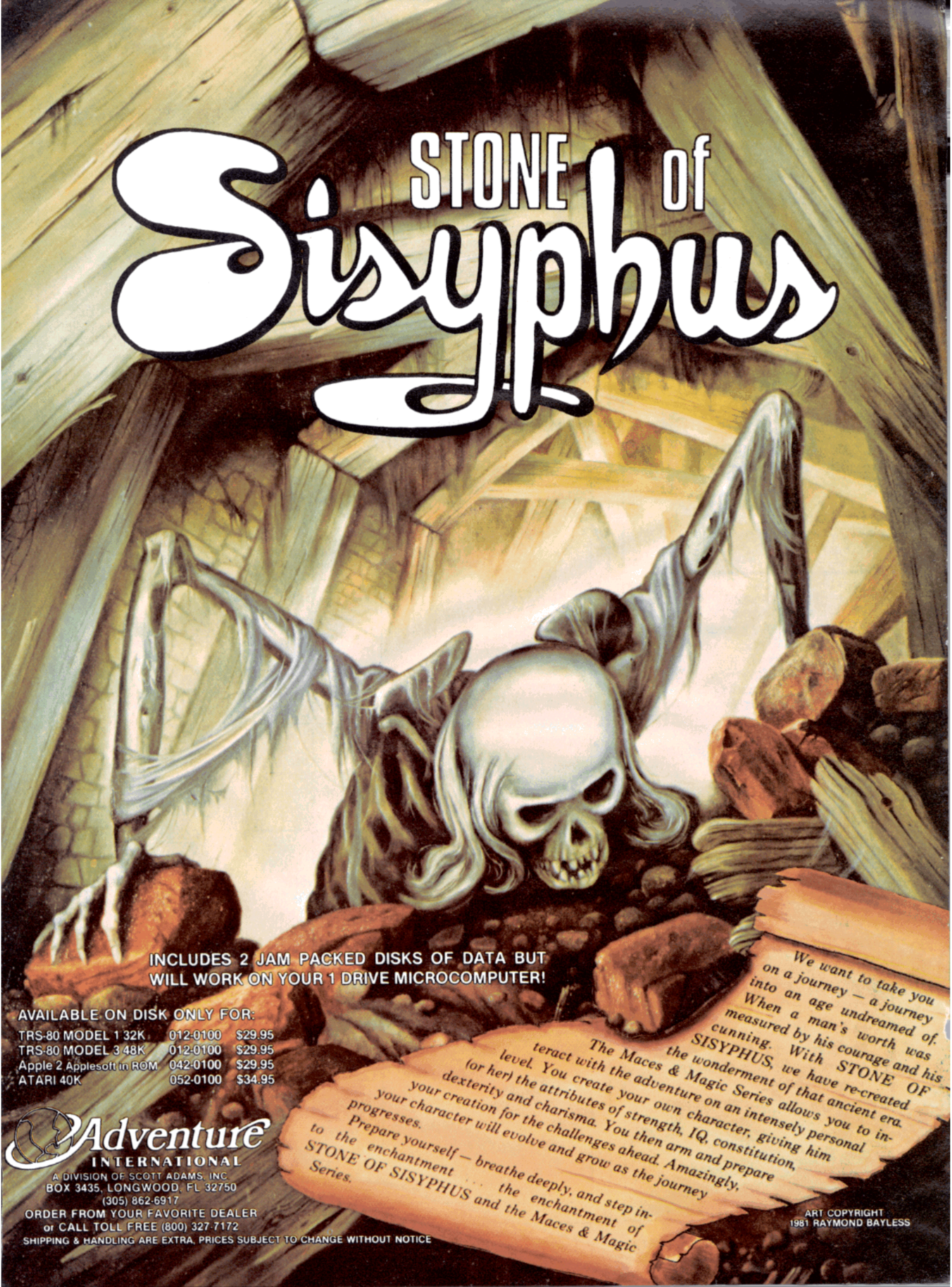


Figure 5. Stock Area

←Reader Serivce for facing page ✓222→


STONE of
Sisyphus
INCLUDES 2 JAM PACKED DISKS OF DATA BUT
WILL WORK ON YOUR 1 DRIVE MICROCOMPUTER!
AVAILABLE ON DISK ONLY FOR:
TRS-80 MODEL 1 32K 012-0100 $29.95
TRS-80 MODEL 3 48K 012-0100 $29.95
Apple 2 Applesoft in ROM 042-0100 $29.95
ATARI 40K 052-0100 $34.95
We want to take you on a journey — a journey into an age undreamed of. When a man's worth was measured by his courage and his cunning. With STONE OF SISYPHUS, we have re-created the wonderment of that ancient era.
The Maces & Magic Series allows you to interact with the adventure on an intensely personal level. You create your own character, giving him (or her) the attributes of strength, IQ, constitution, dexterity and charisma. You then arm and prepare your creation for the challenges ahead. Amazingly, your character will evolve and grow as the journey progresses.
Prepare yourself — breathe deeply, and step into the enchantment . . . the enchantment of STONE OF SISYPHUS and the Maces & Magic Series.
Adventure
INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.
BOX 3435, LONGWOOD, FL 32750
(305) 862-6917
ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172
SHIPPING & HANDLING ARE EXTRA, PRICES SUBJECT TO CHANGE WITHOUT NOTICE
ART COPYRIGHT
1981 RAYMOND BAYLESS

INTERFACE EXPANSION BOARD

was zero; if the result was not zero, the next instruction after the BEQ is executed.

On to the Six Pages

Well, we're finally there: The first machine language related Basic command to consider is Clear. The format of Clear is: CLEAR XX,YYYYY.

The XX represents the decimal number of bytes to reserve for string storage. The YYYYY represents the area of memory to be reserved for machine language programs. Typically, a "Clear 100,YYYYY" would be fine, unless your program has a lot of string storage. The YYYYY is a number that protects RAM beyond a certain point. For our purposes we'll use a YYYYY of $2000, which is 8191 decimal.

This will prevent Basic from using anything above memory location 8191 for stack or string storage. This leaves about 379 bytes of memory for Basic programs and variables, which is somewhat on the sparse side but enough for our purposes.

A memory map for the Color Computer is shown in Fig. 6. Note that Color Basic is above the Extended Color Basic, that video memory starts at $400, and that page 0, 1, 2, and 3 is used for variable storage. Also interesting is the fact that the Color Computer uses high-memory addresses to address various functions for graphics functions, sound generation, serial data, and the like.

Stack Area

We won't explicitly set the S register to

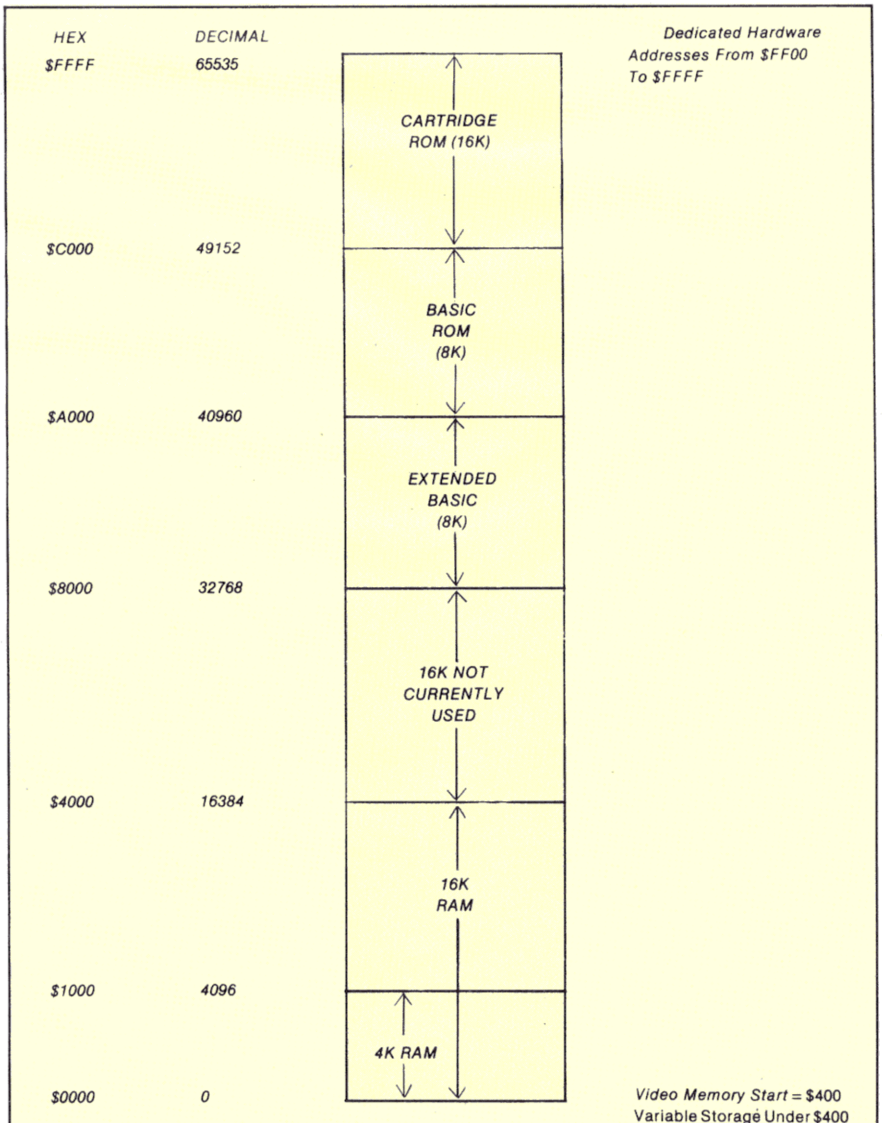


Fig. 6. Color Computer Memory Map

a stack area in the following examples. It is set in Basic, and points to a 30-byte stack area. Unless you're doing a lot of processing, it's not necessary to redefine it by a load of the stack pointer in your program.

The DEFUSRn Command

The next machine language related Basic command we'll tackle is DEFUSRn. DEFUSRn is unsophisticated; it simply tells the Basic interpreter where a user machine language subroutine will be located. The n may be any decimal value from zero through nine, allowing for 10 unique machine language subroutines.

Suppose we had a machine language subroutine for division at location \$2000 (8192 decimal) and another to find the integer solutions to $A^N + B^N = C^N$ (see Fermat's last theorem) at location \$2100 (8448 decimal). We could define the locations by: 100 DEFUSR0 = &H2000: DEFUSR1 = &H2100.

"I was amused by the Color Computer with legs, a result of inbreeding . . ."

From that point on in the Basic program, the ID number of zero would be associated with the divide routine and the ID number of one with the Fermat processing. The ID numbers would be used in conjunction with a USRn call to call the subroutines as required from the Basic program. For example, 1300 A = USR1 (0) would call the Fermat processing routine. The machine code would be entered by USR1 in statement 1300, and, after processing was done, a return would be automatically made to the Basic statement following line 1300.

This is probably a good point to determine what type of machine language subroutines should be called: any type. Whatever proves useful from a Basic program—subroutines that are often called and that can do things faster than the equivalent Basic code, such as sorts of strings, serial I/O, or number crunching. Furthermore, the subroutines may be any size, from two or three instructions, up to thousands. In addition, the subroutines may even be ROM subroutines

THE ASSEMBLY LINE

in Basic ROM that are convenient. One example of this is the cassette subroutines in ROM that allow the user to construct his own cassette tape files. Larger programs that operate only in machine language can be loaded and executed by the CLOADM command, followed by EXEC. We'll give these only a passing mention in this article, as they are usually large dedicated assembly language programs.

The Simplest USR Call

We're finally at a "real" USR call. The Basic code shown in Program Listing 1 defines the simplest call to a machine language subroutine. The subroutine in this case consists of a single instruction, an RTS, or "Return From Subroutine," shown in Fig. 7. The RTS must terminate every machine language subroutine. It POPs the return address of the Basic USR processing code from the hardware stack and causes a return to Basic. All we've really done here is define a machine language subroutine at $A936 and call it by the USR0. The subroutine is repeatedly called by a GOTO 200.

The call was made with two dummy arguments, zero and A. The argument of zero within the parentheses is the input argument. A pointer to its location is passed to the subroutine. The argument of A is the output argument. A possible value from the subroutine is returned in variable A. Both these arguments are dummies in this case, as the subroutine does not require an operand to be passed to it; neither does it return an operand. Use zero for the input and any variable for the output argument as dummies; if you don't, you'll get a nasty message from the Basic interpreter.

Multiple USRn Calls

Next, we'll tackle two USRn calls. Program Listing 2 is a Basic program that defines two machine language subroutines, one at $A936 and another at $A7D7. Both are one-instruction subroutines consisting of RTS instructions in ROM. Here again, the locations are defined, and repetitive calls are made to each subroutine.

A ROM Subroutine Call

Program Listing 3 shows a call to a ROM subroutine at location $A928. From a disassembly, I found that this code clears the screen. The call with two dummy arguments repetitively executes the machine language code to clear the screen.

Passing Arguments to the Subroutine

Those of you making snide noises about simple writers, you're about to get your comeuppance . . .

Let's look at how an argument is passed to a machine language subroutine. Two sentences in the Extended Color manual have implications that Adam Osbourne couldn't explain even

```
100  DEFUSR0 = &HA936
200  A = USR0(0)
300  I = I + 1
400  PRINT I
500  GOTO 200
```

Program Listing 1. Simplest USRn Call.

```
100  DEFUSR0 = &HA928
200  A = USR0(0)
300  GOTO 200
```

Program Listing 2. Calling a ROM Subroutine

```
A931   8C   05FF   CMPX   #1535
A934   23   F9     BLS    $A92F

A936   39          RTS
```

Single RTS
Instruction
At Location
$A936

Fig. 7. Simplest Subroutine

Code	Hex Machine Code	Decimal Machine Code
STA $2000	B7 2000	183,32,0,
STX $2001	BF 2001	191,32,1,
RTS	39	57

Fig. 8. Store A/X Program

```
100  DEFUSR0 = &HA936
150  DEFUSR1 = &HA7D7
200  A = USR0(0)
250  A = USR1(0)
300  I = I + 1
400  PRINT I
500  GOTO 200
```

Program Listing 3. Multiple USRn Calls

with timing charts . . .

According to the manual, the USRn call results in the A register being loaded with a code for the type of argument and the X register being loaded with a pointer to something called a "floating-point accumulator." What does this mean? How did we suddenly go from no arguments to floating point?

To see what is happening, we'll use our first hand-assembled program, shown in Fig. 8. This three-instruction program stores the contents of the A register into location $2000 and the contents of the X register into locations $2001 and $2002. The last instruction is the ubiquitous RTS. We can use this gem to see how A and X look upon entry to the machine language program. The Basic program in Program Listing 4 shows the approach.

The values for the three instructions are first converted to decimal, a total of seven bytes. The code will occupy RAM locations $2003 through $2009. The Basic statements use a data list of the values, a Read and a POKE to move the values from the data list to RAM. Next, a call is made by USR0(0). This call executes the three instructions, which store A and X. The last three Basic statements print location $2000 (A), and locations $2001 and $2002 (X).

When this program is run, we see a display of 0, 0, and 79, indicating that the A register held 0 on entry and that X held 0,79, or 79. (The first value for X is the upper eight bits, while the second is the lower eight bits; together they constitute the entire 16 bits of X.)

The 0 in A indicates a numeric argument, according to the Extended Basic manual. The 79 is a "pointer to the Floating-Point Accumulator which contains the argument." What's in location 79 and what is its format?

We'll answer these questions and more when we continue with the "Saga of the Six Pages" in next month's column. Stay tuned to *80 Micro.* ■

```
100  DATA 183,32,0,191,32,1,57
105  DEFUSR0 = &H2003
110  FOR I = &H2003 TO &H2009
120  READ A
130  POKE I,A
140  NEXT I
150  A = USR(0)
160  FOR I = &H2000 TO &H2002
170  PRINT PEEK(I)
180  NEXT I
```

Program Listing 4. Basic Program to Analyze A/X

EDUCATION 80

"Most of us have wasted a lot of time reloading programs because we did not follow instructions."

How many times have your students loaded programs and had them crash because they forgot to set the memory size? How many times have you done it? I suspect that most of us have wasted a lot of time reloading programs because we did not follow instructions carefully.

All these crashes and reloads are unnecessary. The program itself can set the memory size as soon as it is run. Unfortunately, many programmers have not learned how it is done.

If you are not putting this feature in your programs, these directions will show you how to do so. You can even go back and add "auto mem set" to your old programs and to the programs of others, as well.

Auto Mem Set

This technique works in Level II and in Model III Basic. I have also used it in two versions of TRSDOS and suspect that it will work in all versions. If you use another DOS, try it carefully before you change all your programs just in case your DOS modifies the memory locations used.

Memory locations 16561 and 16562 contain the numbers which protect memory; that is, they set the memory size. All the program has to do is POKE the proper numbers into those places. Location 16561 holds the low-order byte and 16562, the high-order byte. Let's run through a specific example.

Suppose your program requires that memory size be set at 32699. You would go through the following steps:

1. Subtract two from the required size and call the result X.

   32699 − 2 = 32697
   X = 32697

2. Divide X by 256 and discard any remainder (use only the whole number part of the answer). This is the high-order byte.

   32697 / 256 = 127.723
   High Order Byte = 127

3. Multiply the high order byte by 256 and subtract the result from X. This is the low-order byte.

   32697 − (127 × 256) = 185
   Low Order Byte = 185

4. Insert the following statement at the beginning of your program:

```
3 POKE 16561, 185 : POKE 16562, 127
```

By following these steps, the memory size is set at 32699 just as though you had answered the prompt message with that number. Of course, the memory is not set until the program is run.

Did you notice the low line number in step four? It was made low for a reason. The memory size should be set at the beginning of the program—before any other statements are executed (except for remarks). If you set (or change) the memory size after the program is under way, the memory allocation will be wrong for program operation. What happens is that you might jump into areas that the program has set aside for the stacks or variable storage or something else. This would cause the program to crash.

"You might jump into areas the program has set aside. . ."

So, set the memory size first. The line number above 3 will remind you to do that. I would have numbered it "1" but most of us use the first line or two for remarks which identify the program.

Before you change your programs over, this technique may save you some time, too. Suppose you (or a student) begin loading a program and suddenly realize that the memory should have been set. All you have to do is to figure the high and low bytes as above and then execute the statement in step four in the immediate mode before the program is run. The size will be set just as though the statement were in the program.

Chain-Loading Programs

At this time of the year both students and faculty are settled down from the initial flurry of getting school started. This is the time, too, when it dawns on other students and faculty that it might be prudent to learn something about your computer operation.

Often their first question is, "What is the thing good for?" You should have a special program sampler prepared just for such occasions. Sampler—that's what I call a program that consists of several demonstration programs. Here is how you can make one.

The idea is to select a few programs which show off your computer. This might include games, graphic displays or instructional programs.

Next, write a Menu Program which will fill two functions: it allows the operator to choose among the programs you selected, and then it automatically calls up the selected program.

Finally, put all the programs on the tape or disk with the Menu Program and you are ready to go. Let's look at the make-up of that first program.

The Menu Program

The Menu Program is straightforward. It presents a list of the available programs and allows the operator to select one by entering the corresponding number. There are several ways to get the selected program into the computer but I will show you the simplest method. If you use the standard cassette, it would go something like this:

```
....list and selection....
....number selected is in variable A....
250 ON A GOTO 260, 270, 280, 290, 300, etc
260 CLOAD "A"
270 CLOAD "B"
280 CLOAD "C"
290 ...etc...
```

It is evident that programs A, B, C and so on must be properly labeled when saved and they must follow the Menu Program on the tape. Then, too, the tape must be rewound after each selected program has been loaded.

If you are using the Exatron Stringy-Floppy, the appropriate statement would be: 250 @LOAD A + 1. The advantage to the ESF is its fast load time and the fact that it does not have to be rewound. However, you should provide enough space to hold the longest program (see the manual for details).

If you are using Disk Basic, your statements would look like this:

```
250 ON A GOTO 260, 270, 280,......
260 RUN "STORYBOOK"
270 RUN "MATH I"
280 .....
```

Of course, both the cassette and disk systems offer neater ways to accomplish this programming, but these are the easiest to understand. You might experiment to discover shorter methods with your system.

I have put together several disks which follow this plan—one for science, one for math, one for language arts, and so on. Since making them, life has been easier. I simply put in the appropriate disk and say, "Try it for yourself."

Reader Inquiry

Bjarne Madsen in Saksatchewan, Canada, leads a project which developed a curriculum for computer awareness and programming techniques at the junior high school level. His course has proved very successful in classroom tests.

BJ is quite interested in corresponding with others who may be involved in a similar effort. The exchange of ideas could be beneficial to both parties. If you have worked in this area, send me your name and I'll pass it along to BJ.

In fact, if you are interested in communicating with others in any area of interest, send me your name, complete address, and the areas of interest. Include a statement to the effect that you would like that information published and, as space permits, I'll put it in Education 80 so that others can get in touch with you.

New Address

This summer I took time out to move across the state. For six weeks or so I was out of touch with everything except boxes to pack and unpack. I don't know if I'll ever catch up with what happened in the world of computers. (I have caught up with the new Model III TRSDOS 1.3 which offers significant improvements on an already good 1.2 version.)

However, I am happy to report that my computers made the trip in fine shape—not even a slight hiccup when I finally got settled enough to hook all the pieces together and plug them in. So much for my fears of horrendous repair bills!

My new address is 6 Darl Avenue, Colonial Beach, VA 22443. Of course, you may write me through *80 Microcomputing*. Either way, please enclose a self-addressed, stamped envelope and I'll get back to you when time permits. ■

✓See List of Advertisers on page 386

80 APPLICATIONS

by Dennis Kitsz

Way back in the early days of *80 Microcomputing,* I created a machine-language monitor called Babybug. It was short, written in Basic, and sparked with a large and continuing reader response. Since that time, many excellent programs for examining and modifying TRS-80 memory have appeared, and some interesting software is beginning to be offered for the Model III and the Color Computer as well.

This month I'd like to present new versions of the original Babybug: a condensed and improved Model I or III program, plus versions for Color Basic and Extended Color Basic. Once again, they are written entirely in Basic, yet consume small amounts of memory. Before turning to the programs, I'd like to explain what a machine language monitor is, how it is used, talk about entry and display in hexadecimal numbering, and in passing describe the process of packing information into integers.

A Window into Memory

At some point in the life of every programmer—amateur or professional—comes the desire to have programs perform faster or use less memory. There also comes creeping the need to understand what makes the computer tick, why protected memory is sometimes necessary, and maybe just what kind of software is hidden inside that anonymous box. It's that Something which gives the system life, makes it say "OK" when it understands you and "?SN ERROR" when it doesn't.

Small computers are made up of several classes of black boxes which compose a computer when connected together. In its purest form, the computer we know consists of a general-purpose, simple-minded calculator—the central processing unit (CPU). The CPU is connected to one or more blocks of temporary or fixed storage/instruction area, its memory. Finally, communication with the real world is achieved with input/output (I/O) devices.

The Babybug machine-language monitor is concerned with the contents of read/write memory—that is, memory which may be examined or changed by the CPU. It can also be used to examine permanent (read-only) memory, in order to see what the CPU follows to perform its work in Basic. With a monitor, I can ask the computer to tell me what is found at its memory location 0 or 100 or 24576 or 65500. In the Model I, if I examine location 0 I will find the value 243; if I examine location 100, chances are the value 251 will turn up. Both 0 and 100 are in permanent, read-only memory (ROM). On the other hand, locations 0 and 100 are in temporary memory in the Color Computer, holding the values 0 and 171 when Basic is in operation. Model I locations 24576 or 65500 may reveal anything, since they are read/write memory (also called random-access memory, or RAM)—whatever the computer last placed there. In the Color Computer, 24576 is read-only memory, and 65500 is part of I/O memory. With the Babybug machine-language monitor, I can read memory and change its contents at will.

What is the machine language that Babybug monitors? Since the information stored in any spot in memory can be thought of as instructions intended for the CPU's internal mechanism, causing it to respond according to a predetermined pattern, such memory contents are called machine language, or machine code. Throughout this magazine, you will find listings written in Basic, but other software is presented in rigorous-looking listings for the Editor/Assembler. This is assembly language, merely a convenient form of producing numerical machine instructions using English-sounding names instead of numbers alone.

The basics of machine and assembly programming have been presented in this column before (see especially March 1981), in Bill Barden's monthly Assembly Line column, as well as in many other fine articles. These fundamentals won't be repeated here, but I would like to explain why so often you will find machine code used in this column, and why nondecimal notation can be very revealing when programming at the machine level—as well as in sophisticated Basic.

Computers are nothing more than an array of fast on-off toggle switches, which means if you learn to read the toggle switches, you know exactly what your computer is doing. For example, in May I presented a Micro Front Panel construction project. It's a bank of 24 LEDs which flash a binary value according to the current memory location the computer is using. It displays both the location (address) and the information (data). Several people have written to me asking what good it is, and I reply with an example: when programs seem to get lost, there's practically no way to know whether they are truly lost, merely involved in some time-consuming process, or hung up in a minor program glitch.

Something peculiar happened when I tried to load one disk. The familiar noises took place, but then everything just stopped. I pressed the reset button again, the system reacted normally, but then just stopped. I glanced down at my Micro Front Panel, which read:

LSB	MSB	DATA
0000 0000	0100 0010	0000 0000

It told me that my computer stopped writing data at binary address 0100 0010 0000 0000 (which translates to hexadecimal address 4200) and the data it wrote there was 00. I happen to know that 4200 is where the disk's bootstrap load routine is placed in memory, so I learned that the bootstrap loader was never written.

With the front panel I can also observe that the machine is still working when it pauses during a long program and will not respond to the Break key. Those of you whose programs have been trapped in seemingly interminable string sorts (where garbage collection is in progress) can be reassured that there is activity even though the machine seems hung.

My point? Only that there is no way to know exactly where a machine is hung (and what to do about it) unless you can decipher binary numbers.

If that's too esoteric a notion, try this simpler one from the realm of Basic: By using a *single integer,* you want to find out whether a person on your mailing list (a) has one of up to eight special skills; (b) was originally contacted through one of eight possible methods; (c) responded to

one of eight mailings; (d) attended the last meeting; (e) was contacted by phone; (f) paid the most recent dues; (g) registered for an upcoming seminar; (h) participated in an event; and (i) turned in a survey form. Can you do it? In binary, it's possible.

Data Packing is Binary

Program Listing 1 is a possible solution to that question, drawn from a larger program. It requires an understanding of the binary consequences of some integer action in the computer, and also demands knowledge of the logical functions (AND, OR). Although this section of the program by itself seems lengthy, the resulting information is only *one* integer, meaning it can be stored in your mailing list or other data file as an economical single integer—just two bytes of memory or file space!

Here's a look at the byte the program starts with, represented by integer Q: 0000 0000 0000 0000. In the TRS-80, the leftmost binary digit (bit) is reserved for the number's sign. Zero defines a positive integer, one defines a negative integer. Since coded information containing a minus sign might look strange, I've discarded the leftmost bit in this scheme, leaving: (0)000 0000 0000 0000.

Defining the arrangement of information to be stored in the two-byte integer is arbitrary. In the sample listing, there are eight choices for the first input (lines 150 to 280 in Listing 1), so I reserved the leftmost three bits for the response. Why only three bits for eight choices? Because there are eight possible combinations of these three bits: 000, 001, 010, 011, 100, 101, 110, and 111. Likewise, the second input (lines 310 to 390) can allow eight possible choices, as can the third input (lines 420 to 550). The remaining questions are yes or no responses, meaning only a single bit need be used for each (1 = yes, 0 = no). The result is a positive integer with its bits divided as in Table 1.

Now, the bits themselves have to be made accessible to us, which means it's essential to know the value of each bit in the integer. Since this is binary numbering, these are simply powers of two. Simply, I said? Sure. In decimal, the numbers are ranked according to powers of 10, which in elementary school was referred to as the "one's place," the "ten's place," the "hundred's place," etc. In binary, the numbers are still called out by their places, except that they would be called the "one's place," "two's place," "four's place," "eight's place," and so on.

That breaks the binary integer up as shown in Table 2.

✓See List of Advertisers on page 386

Turn to line 300 in Listing 1, particularly the command A = (A-1) × 4096. Notice that the variable A was set to be the numerical result of the input from choices one through eight, in lines 170 to 240. Since the bit patterns to be used to represent these choices run from 000 to 111 (decimal 0 to 7), the value of A must be decreased by one to force it to be within the range 0 to 7. That's the easy part.

The upper three bits of the integer are going to be used to store variable A; those upper three bits are the 16384's place, the 8192's place, and the 4096's place. Table 3 is a chart of the possible combinations.

With this in mind, you can see how line 300 creates the upper part of the integer out of this pattern of eight choices by multiplying by 4096. The final command in the line is Q = Q OR A. Again, this is visualized in binary, since logical functions AND, OR, and NOT operate on binary digits. Here's what happens, assuming choice 8 was selected:

Q at start	=	000 0000 0000 0000
A is selected	=	111 0000 0000 0000
Q OR A function		----------------------------------
Result	=	111 0000 0000 0000

Remember that the OR function specifies that if either bit X or bit Y (or both) is 1, then the result will be 1. The reason for using the OR function here is to leave the original bits untouched while setting (changing to 1) the bits that are needed to produce a packed byte. Turn to line 410 to see how another group of bits is packed into integer Q. Again, the first part of the command A = (A-1) × 512 moves variable A from its original input range of 1 to 8 down to the range of 0 to 7, because three bits are being reserved in the integer byte (000 to 111) with decimal values from 0 to 7.

The response bits for this input fall (referring to the binary "place" chart) in the 512's, 1024's, and 2048's place. Likewise, the patterns appear as in Table 4.

All are multiples of 512, so the command A = (A-1) × 512 sets up the correct bits. In this case, assume that the user's choice was number 6, resulting in a bit pattern of 101 in the appropriate integer's positions. The OR function is used again in line 410, but recall that Q already has been assigned a value in line 300:

Q new value	=	111 0000 0000 0000
A now selected	=	000 1010 0000 0000
Q OR A function		----------------------------------
New Q result	=	111 1010 0000 0000

What is the resulting Q at this point? You can calculate it by adding the sum of the powers of two, as above, and you will discover the integer is now 31232. Or, more simply, you can hit the Break key and enter the command: PRINT Q. 31232 will be the result.

The yes or no questions are simpler. If the answer to a question is yes, its respective bit is set to 1. If the answer is no, that bit is set to 0. The answers to the six questions in this example are packed into integer Q in lines 610-620, 660-670, 710-720, 860-770,810-820, and 860-870. For example, assuming a yes answer to question 4, variable A would be set to 32, then ORed with Q. Notice that if you wish to pack the answers to 15 yes or no questions into a single integer (instead of multiple-selection responses), this method can be used easily.

The final three bits of integer Q hold the results of another eight-answer ques-

```
10 REM * THIS ROUTINE CAN BE REVISED
20 REM * TO INCLUDE GENERAL-PURPOSE
30 REM * CATEGORIES IN PLACE OF THE
40 REM * SPECIFIC ONES SHOWN.  UP TO
50 REM * 15 YES/NO QUESTIONS CAN BE
60 REM * STORED, OR SEVEN 4-CHOICE
70 REM * QUESTIONS.  A CASSETTE OF
80 REM * THIS DEMO AND ALL THE BABYBUG
90 REM * PROGRAMS IS AVAILABLE FROM
100 REM * THE AUTHOR, ROXBURY VT 05669
110 CLS:PRINT"***** STRING PACKING DEMO *****"
120 PRINT"TOUCH 1 TO UPDATE,  2 TO DECODE"
130 PRINT STRING$(31,42)
140 A$=INKEY$ : IF A$="1" THEN 150 ELSE
     IF A$="2" THEN 1340 ELSE 140
150 GOSUB 1380
160 PRINT "FIRST CONTACT WAS:"
170 PRINT "1 = FRIEND"
180 PRINT "2 = PROFESSIONAL ASSOCIATE"
190 PRINT "3 = ARTS ORGANIZER"
200 PRINT "4 = FESTIVAL PARTICIPANT"
210 PRINT "5 = FESTIVAL ATTENDEE"
220 PRINT "6 = BENEFIT CONTRIBUTOR"
230 PRINT "7 = MISCELLANEOUS"
240 PRINT "8 = <RESERVED>"
250 PRINT STRING$(27,45)
260 PRINT "TOUCH APPROPRIATE NUMBER."
270 PRINT STRING$(27,45)
280 A$=INKEY$ : A=VAL(A$) : IF A<1
     OR A>8 THEN 280
290 REM * FIRST BIT PACKING HERE
300 Q=0 : A=(A-1)*4096 : Q=Q OR A
310 GOSUB 1380
320 PRINT "MAILING RESPONSE."
330 PRINT "MAILINGS TO DATE:"
340 PRINT "    -  6  -     "
350 PRINT "HOW MANY RESPONSES?"
360 PRINT STRING$(27,45)
370 PRINT "TOUCH APPROPRIATE NUMBER."
380 PRINT STRING$(27,45)
390 A$=INKEY$ : A=VAL(A$) : IF A<1
     OR A>8 THEN 390
400 REM * SECOND BIT PACKING HERE
410 A=(A-1)*512 : Q=Q OR A
420 GOSUB 1380
430 PRINT "ARTISTIC CATEGORY:"
440 PRINT "1 = VISUAL ARTIST"
450 PRINT "2 = DESIGNER"
460 PRINT "3 = COMPOSER"
470 PRINT "4 = VIDEO ARTIST"
480 PRINT "5 = DANCER/CHOREO"
490 PRINT "6 = SCULPTOR"
500 PRINT "7 = PERF. MUSICIAN"
510 PRINT "8 = OTHER"
520 PRINT STRING$(27,45)
530 PRINT "TOUCH APPROPRIATE NUMBER."
540 PRINT STRING$(27,45)
550 A$=INKEY$ : A=VAL(A$) : IF A<1
     OR A>8 THEN 550
560 REM * THIRD BIT PACKING HERE
570 A=A-1 : Q=Q OR A
580 GOSUB 1380
590 PRINT "QUESTION #1***"
600 GOSUB 1420
610 A$=INKEY$ : IF A$="Y" THEN A=256
     ELSE IF A$="N" THEN A=0 ELSE
     610
620 Q=Q OR A
630 GOSUB 1380
640 PRINT "QUESTION #2***"
650 GOSUB 1420
660 A$=INKEY$ : IF A$="Y" THEN A=128
     ELSE IF A$="N" THEN A=0 ELSE
     660
670 Q=Q OR A
680 GOSUB 1380
690 PRINT "QUESTION #3***"
700 GOSUB 1420
710 A$=INKEY$ : IF A$="Y" THEN A=64
     ELSE IF A$="N" THEN A=0 ELSE
     710
720 Q=Q OR A
730 GOSUB 1380
740 PRINT "QUESTION #4***"
750 GOSUB 1420
760 A$=INKEY$ : IF A$="Y" THEN A=32
     ELSE IF A$="N" THEN A=0 ELSE
     760
770 Q=Q OR A
780 GOSUB 1380
790 PRINT "QUESTION #5***"
800 GOSUB 1420
810 A$=INKEY$ : IF A$="Y" THEN A=16
     ELSE IF A$="N" THEN A=0 ELSE
     810
820 Q=Q OR A
830 GOSUB 1380
840 PRINT "QUESTION #6***"
850 GOSUB 1420
860 A$=INKEY$ : IF A$="Y" THEN A=8
     ELSE IF A$="N" THEN A=0 ELSE
     860
870 Q=Q OR A
880 GOSUB 1380
890 PRINT "INPUT PHASE IS COMPLETE."
900 PRINT "CODE NUMBER = ";
910 Q$=RIGHT$(STR$(Q),LEN(STR$(Q))-1)
920 PRINTSTRING$(5-LEN(Q$),"0");Q$"."
930 PRINT "DECODING INFORMATION:"
940 R=Q : R=R AND 28672 : R=R/4096 :
     R=R+1
950 PRINT "CONTACT: ";
960 ON R GOTO 970,980,990,1000,1010,
     1020,1030,1040
970 PRINT "FRIEND" : GOTO 1050
980 PRINT "PROFESSIONAL ASSOCIATE":
     GOTO 1050
990 PRINT "ARTS ORGANIZER" : GOTO
     1050
1000 PRINT "FESTIVAL PARTICIPANT" :
     GOTO 1050
1010 PRINT "FESTIVAL ATTENDEE" :
     GOTO 1050
1020 PRINT "BENEFIT CONTRIBUTOR" :
     GOTO 1050
1030 PRINT "MISCELLANEOUS" : GOTO
     1050
1040 PRINT "<RESERVED>"
1050 PRINT "MAILING RESPONSES:";
1060 R=Q : R=R AND 3584 : R=R/512 :
     R=R+1 : PRINTR
1070 PRINT "CATEGORY: ";
1080 R=Q : R=R AND 7 : R=R+1
1090 ON R GOTO 1100,1110,1120,1130,
     1140,1150,1160,1170
1100 PRINT "VISUAL ARTIST" : GOTO
     1180
1110 PRINT "DESIGNER" : GOTO 1180
1120 PRINT "COMPOSER" : GOTO 1180
1130 PRINT "VIDEO ARTIST" : GOTO
     1180
1140 PRINT "DANCER/CHOREO" : GOTO
     1180
1150 PRINT "SCULPTOR" : GOTO 1180
1160 PRINT "PERF. MUSICIAN" : GOTO
     1180
1170 PRINT "UNCATEGORIZED" : GOTO
     1180
1180 PRINT "QUESTION #1: ";
1190 R=Q : R=R AND 256 : IF R=256
     THEN PRINT "Y" ELSE PRINT "N"
1200 PRINT "QUESTION #2: ";
1210 R=Q : R=R AND 128 : IF R=128
     THEN PRINT "Y" ELSE PRINT "N"
1220 PRINT "QUESTION #3: ";
1230 R=Q : R=R AND 64 : IF R=64
     THEN PRINT "Y" ELSE PRINT "N"
1240 PRINT "QUESTION #4: ";
1250 R=Q : R=R AND 32 : IF R=32
     THEN PRINT "Y" ELSE PRINT "N"
1260 PRINT "QUESTION #5: ";
1270 R=Q : R=R AND 16 : IF R=16
     THEN PRINT "Y" ELSE PRINT "N"
1280 PRINT "QUESTION #6: ";
1290 R=Q : R=R AND 8 : IF R=8 THEN
     PRINT "Y" ELSE PRINT "N"
1300 PRINT STRING$(27,45);
1310 FOR X=1 TO 1000 : NEXT
1320 PRINT@64, "PRESS <ENTER>          ";
1330 A$=INKEY$ : IF A$=CHR$(13) THEN
     RUN ELSE 1330
1340 CLS : PRINT STRING$(27,45)
1350 INPUT "CODE NUMBER";Q
1360 PRINT STRING$(27,45)
1370 GOTO 900
1380 CLS : PRINT STRING$(27,45)
1390 PRINT "     ** LIST UPDATE **"
1400 PRINT STRING$(27,45)
1410 RETURN
1420 PRINT STRING$(27,45)
1430 PRINT "TOUCH <Y>ES OR <N>O."
1440 PRINT STRING$(27,45)
1450 RETURN
```

Program Listing 1

✓ See List of Advertisers on page 386

tion, and the process (which I won't describe this time) is carried out in lines 420 through 570 of Listing 1.

Getting It Back

Once the information has been packed into integer Q (which will be a number from 0 to 32767), how is it retrieved? Lines 900–920 print the number on the screen in five-digit format, merely for convenience. Its decoding follows, employing another logical function: AND.

In brief, if bit X is 1 and bit Y is 1, then the result of the function X AND Y will be 1. Any other combination of X AND Y will result in a 0. Look at line 940, which uses the AND function in the command R = Q : R = R AND 28672. Remember that at this point Q is the coded integer containing a lot of information; by making R equivalent to Q, value Q can be saved intact for later use. So this value R (= Q) is ANDed with 28672. What is 28672? It is 16384 + 8192 + 4096. In other words, the value 28672 is an integer with the upper three bits already set to ones (111 0000 0000 0000). This is the key to recovering the information packed into Q.

The following example shows a possible coded number Q (decimal 22190) generated with this packing system. It is being ANDed with 28672, which will "mask out" all the bits we do not need to know about. Since everything ANDed with 0 will end up being 0, this method simply places zeros where they are needed in the masking integer:

```
(0)    000    000    0    0    0    0    0    0     000
---    I#1    I#2    Q#1  Q#2  Q#3  Q#4  Q#5  Q#6   I#3
Not    Eight  Eight  Yes  Yes  Yes  Yes  Yes  Yes   Eight
Used   Resp.  Resp.  /No  /No  /No  /No  /No  /No   Resp.
```

Table 1

```
1
6  8  4  2  1
3  1  0  0  0  5  2  1
8  9  9  4  2  1  5  2  6  3  1
4  2  6  8  4  2  6  8  4  2  6  8  4  2  1
'  '  '  '  '  '  '  '  '  '  '  '  '  '  '
s  s  s  s  s  s  s  s  s  s  s  s  s  s  s

P  P  P  P  P  P  P  P  P  P  P  P  P  P  P
l  l  l  l  l  l  l  l  l  l  l  l  l  l  l
a  a  a  a  a  a  a  a  a  a  a  a  a  a  a
c  c  c  c  c  c  c  c  c  c  c  c  c  c  c
e  e  e  e  e  e  e  e  e  e  e  e  e  e  e
```

Table 2

```
Bit Position:  (0)XXX 0000 0000 0000
                  ---

000 =     0 +    0 +    0  =      0  (0 x 4096)
001 =     0 +    0 + 4096  =   4096  (1 x 4096)
010 =     0 + 8192 +    0  =   8192  (2 x 4096)
011 =     0 + 8192 + 4096  =  12288  (3 x 4096)
100 = 16384 +    0 +    0  =  16384  (4 x 4096)
101 = 16384 +    0 + 4096  =  20480  (5 x 4096)
110 = 16384 + 8192 +    0  =  24576  (6 x 4096)
111 = 16384 + 8192 + 4096  =  28672  (7 x 4096)
```

Table 3. Combination Chart

```
Bit Position = (0)000 XXX0 0000 0000
                      ---

000 =    0 +    0 +   0  =     0  (0 x 512)
001 =    0 +    0 + 512  =   512  (1 x 512)
010 =    0 + 1024 +   0  =  1024  (2 x 512)
011 =    0 + 1024 + 512  =  1536  (3 x 512)
100 = 2048 +    0 +   0  =  2048  (4 x 512)
101 = 2048 +    0 + 512  =  2560  (5 x 512)
110 = 2048 + 1024 +   0  =  3072  (6 x 512)
111 = 2048 + 1024 + 512  =  3584  (7 x 512)
```

Table 4

Q integer being tested:	101 0110 1010 1110
28672 used as a mask:	111 0000 0000 0000
Executing Q AND 28672:	------------------------
Resulting masked value:	101 0000 0000 0000

Any 1 bits in integer Q (22190) will filter through the mask, giving a binary result. Converting the result to decimal gives the value 20480 in this example. The next command in line 940 is R = R/4096. Recall that value A in the input phase of the sample program was multiplied by 4096 to shift the bits into the high position of integer Q. Here is the reverse of the process, resulting in an integer with the value 0 through 7. Since the original choices were 1 through 8, the final command on line 940 is R = R + 1. In this sample, R (= 22190) AND 28672 = 20480. R/4096 = 5. R + 1 = 6. Therefore, the information has been recovered, and it can now be stated that the original value input was sixty. It is only a simple matter (in line 960) to display what that original input choice means.

Line 1060 performs the same sort of activity to recover the response to the second input. Again, R = Q to recover the initial packed integer. Then R = R AND 3584, since 3584 equals 2048 + 1024 + 512 (another three-bit mask). R = R/512 shifts the values to 0 through 7, and R = R + 1 returns the values 1 through 8. Displaying the original input is again simple. Here is the binary representation:

R (= Q):	101 0110 1010 1110
3584 Mask:	000 1110 0000 0000
R AND 3584:	------------------------
Recovered info:	000 0110 0000 0000

When all this information is recovered, it's possible, by feeding in a five-digit code on someone's mailing label, to print out a report for your club or user's group which says, "John Jones is a TRS-80

Model III user with whom we made contact through another member. He has responded to each of our mailings, attended the last meeting, participated in the flea market, his dues are up to date. Jones's specialty is assembly language, and he participated in the club's last seminar." All in one two-byte integer.

How to Hex

Okay, you say, binary has its uses if I'm willing to spend some time and effort to use it. But why hexadecimal notation? Who cares? Hexadecimal notation, as a reflection of its binary function, can clearly demonstrate the relationships between machine functions. A very few of the 6809E instructions, commands which affect the operation of the Color Computer's central processing unit, are in Table 5.

Obviously, the easiest way to remember these instructions is by their mnemonics. But the TRS-80 cannot easily report back its memory contents in mnemonics, at least not without an extensive programmer's aid called a disassembler. Therefore, when you see a bunch of numbers, ideally they should provoke some familiar reaction. Numbers like 136, 152, 168, and 184 are not likely to emphasize their straightforward relationship to each other. Examine instead Table 6, with those same four instructions (all sixteen forms) shown in mnemonics, decimal, binary, and hexadecimal.

See how neatly the leftmost digits count upward in binary? And notice how the commands count upward, in hex 10's, in that same neat way. The relationships between the versions of ADC are visible, as are the relationships between the variants of EOR. Their nature is clearest in binary, but it is also easily visible in hexadecimal—a far cry from attempting to quickly discern the relationship between 136 and 152, or 200 and 232.

Not impressed? Then how about an excerpt from Bill Barden's August 1981 Assembly Line. On page 49, a man with staggering insight named David Lamkins offered this remarkable binary-to-ASCII conversion routine (wish *I'd* thought of this one):

```
ADD   A,90H
DAA
ADC   A,40H
DAA
```

Let's try it first in decimal to see if it

Command Mnemonic	Description of Command's Action	Decimal Values
ADCA	Add to accumulator A, plus carry (instruction has four options)	137,153, 169,185
ADCB	Add to accumulator B, plus carry (instruction has four options)	201,217, 233,249
EORA	Exclusive OR accumulator A (instruction has four options)	136,152, 168,184
EORB	Exclusive OR accumulator B (instruction has four options)	200,216, 232,248

Table 5. 6809E Instructions

Mnemonic	Decimal Value	Binary Value	Hex Value
ADCA	137	1000 1001	89
	153	1001 1001	99
	169	1010 1001	A9
	185	1011 1001	B9
ADCB	201	1100 1001	C9
	217	1101 1001	D9
	233	1110 1001	E9
	249	1111 1001	F9
EORA	136	1000 1000	88
	152	1001 1000	98
	168	1010 1000	A8
	184	1011 1000	B8
EORB	200	1100 1000	C8
	216	1101 1000	D8
	232	1110 1000	E8
	248	1111 1000	F8

Table 6

makes sense. Add 144 to the accumulator, decimal adjust the accumulator (for base ten operations), add 64 to the accumulator plus the contents of the carry flag, and decimal adjust the accumulator again. Hmm. Let's convert decimal 14 to ASCII with it. 144 + 14 = 158. Decimal adjust. From what? The Z80 assembly programming guide talks in terms of hex values in upper and lower digits, but I've converted it to decimal equivalents for this example. Take my word that DAA would add 102 to this value, resulting in 260. Since the maximum value is 255, this 260 must have actually produced four and set the carry flag. Add 64, with carry, to the accumulator produces 69—the ASCII code for uppercase letter E.

Was that clear? I really didn't try to murky it up at all; it's just the best that can be done in decimal terms. Now here's the same conversion of 0E (decimal 14) to ASCII using hexadecimal notation. 90H + 0EH = 9EH. Decimal adjust. The programming guide says if 9–F is in the upper digit and A–F is in the lower, decimal adjust (DAA) adds 66H to this result. 9EH + 66H = 104H, with the leftmost 1 ending up in the carry flag. Add 40H, with carry, produces 04H + 40H + 1 = 45H—the ASCII code for letter E.

Any clearer? Obviously, since the programming guide forces you to use hex digits to determine the decimal-adjusted results, that consideration is simpler. But also, notice that decimal 256 is actually 100 hex—in terms of the accumulator, it is 00 with the 1 carried over. That's why 260 decimal produces a result of four with carry, not five. It just isn't visible to the eye in decimal notation, whereas hexadecimal makes it very certain that a carry *has* taken place.

I've jumped to the defense of hexadecimal notation not because I'm a longtime programmer rigidly committed to it (I'm neither), but rather because it's essential to understanding what your computer is doing, if understanding is what you care to do. Sure, numbering systems are arbitrary, and it's unfortunate that we're stuck with binary notation and digital logic to start with. But it's currently an economic fact of life for computers; even if computers move away from binary operation, *your* TRS-80 Model I, II, III, Color Computer, Videotex, Pocket Computer—or your Apple, OSI, Pet, Northstar, Atari, Vic, or Compusquat—is a digital device. That's its nature, so that becomes your problem to deal with in the least time-consuming manner.

What VAR Is that VARPTR PTRing to?

One more example; hang in there with me. Most programmers end up being forced to write sorting programs. Aside from being an unsatisfying project, sorting programs written in Basic can be dreadfully slow. Normally, sorting is done by comparing one item with another in some manner, and switching those items. But if the items are very long (such as names and addresses or other long strings), the sorts can take hours to complete.

There's another way. Imagine you are the curator of a museum full of artwork, each piece with a reference number tacked nearby. You have had the catalog reprinted, but the artwork now does not

match the numbered reference labels. Would you (a) move the artwork to match the labels, or (b) move the labels to match the artwork? Assuming there has been no artistic decision involved in rearranging the catalog, it's probably a lot easier to move a label instead of, perhaps, a two-ton piece of sculpture.

This relates directly to sorting programs. Every variable is located by Basic by means of a variable table it builds and stores just after the program listing. Your computer can provide you with that information by means of the command VARPTR (variable pointer). If variable R$ has been defined, then command PRINT VARPTR(R$) will return the decimal value of the address at which all the information about R$ is stored.

If PRINT VARPTR(R$) yields 8591 on the Color Computer, then PRINT PEEK(8591) will reveal the length of R$, and PRINT 256 × PEEK(8593) + PEEK (8594) will tell where R$ starts in memory. For example, in the Color Basic version of Babybug, the command PRINT VARPTR(R$) might in fact yield 8591, PEEK(8591) = 2, and 256 × PEEK(8593) + PEEK(8594) = 16381. Interestingly, you will find that the formidable question—

```
FOR X = 1 TO PEEK (VARPTR (R$) ) : PRINT CHR$
(PEEK (256 x PEEK (VARPTR (R$) + 2) + PEEK (VARP-
TR (R$) + 3) + X - 1 ) ); : NEXT
```

—will actually print R$ on the Color Computer! On the Model I or III, one aspect would be reversed, and the equation would read:

```
FOR X = 1 TO PEEK (VARPTR (R$) ) : PRINT CHR$ (PEEK
(PEEK (VARPR (R$) + 1) + 256 x PEEK (VARPTR (R$) + 2)
+ X - 1) ); : NEXT
```

What is this 256 × PEEK(N) + PEEK (N + 1) business? Because there are so many memory locations in the computer, any given location must always be specified using sixteen binary digits of data—one group of eight bits defines the most significant byte (MSB), or highest eight binary digits, and the other group defines the least significant byte (LSB) of the address. You will recall that integers were also stored as two bytes of information, so this type of format is not new. What is new, at least in the Model I or III TRS-80s, is how this information is ordered.

Because of choices made when the original 8008 microprocessor was designed in the early 1970's, its grandchild the Z80 stores addresses with the LSB *first* and the MSB *second*. Fortunately, users of the Color Computer will find that the 6809E processor keeps things in a more human-oriented order, with MSB first and LSB second. In both cases, though, the lower eight bits can be used as written, but the upper eight bits occupy the 256's place through the 32768's place. Therefore, the number represented by this byte is actually 256 times that of the least significant byte.

Now to the point. If I PEEK into the Model I's memory and find that an address is given as 233 decimal, 66 decimal, I can calculate the true address with the formula LSB + 256 × MSB, or 233 + 66 × 256. That is 17129, which many of you will recognize as the beginning of Basic program storage. If I PEEK into the Color Computer's memory and find an

address written 30 decimal, 0 decimal, I can calculate that address as 256 × MSB + LSB, or 256 × 30 + 0. That is 7680, also the beginning of Basic.

```
10 REM * THIS PROGRAM DEMONSTRATES
20 REM * A BUBBLE SORT WRITTEN WITH
30 REM * BOTH NORMAL REPLACEMENT
40 REM * SORTING AND WITH VARPTR
50 REM * REFERENCE POINTER SWITCHING
60 CLEAR25Ø:CLS:PRINT STRING$(27,45)
70 PRINT "<N>ORMAL STRING SORT"
80 PRINT "<V>ARPTR STRING SORT"
90 PRINT STRING$(27,45)
1ØØ PRINT "TOUCH N OR V."
11Ø PRINT STRING$(27,45)
12Ø A$=INKEY$ : IF A$="N" THEN 13Ø
    ELSE IF A$="V" THEN 28Ø ELSE
    12Ø
13Ø REM ............................
14Ø REM * BEGIN NORMAL SORT PROCEDURE
15Ø REM ............................
16Ø INPUT"HOW MANY ITEMS TO SORT";I
17Ø DIM A$(I) : FOR X = 1 TO I :
    PRINT X; : INPUT A$(X) : NEXT :
    GOSUB 44Ø
18Ø PRINT "< ... NOW SORTING ... >";
19Ø GOSUB 44Ø : REM REMOVE TO TIME
2ØØ X=1:N=N+1:Q=Ø
21Ø IFA$(X)<=A$(X+1)THEN23Ø
22Ø Q=Q+1:Z$=A$(X):A$(X)=A$(X+1):
    A$(X+1)=Z$
23Ø X=X+1:IFX>I-1THEN24ØELSE21Ø
24Ø IFQ=ØTHEN25ØELSEIFX>I-1THEN19Ø
    ELSE21Ø
25Ø GOSUB 44Ø
26Ø PRINT "<SORTED - TOUCH ENTER>";
27Ø A$=INKEY$ : IF A$=CHR$(13) THEN
    1Ø ELSE 27Ø
28Ø REM ............................
29Ø REM * BEGIN VARPTR SORT PROCEDURE
3ØØ REM ............................
31Ø DEFINTX,Y,Z,I,K,L,M,N,Q,D,E :
    Y=Ø : Z=Ø : D=1 : E=2 : K=Ø :
    L=Ø : M=Ø : N=Ø : Q=Ø : X=Ø
32Ø INPUT"HOW MANY ITEMS TO SORT";I
33Ø DIM A$(I) : FOR X = 1 TO I :
    PRINT X; : INPUT A$(X) : NEXT :
    GOSUB 44Ø
34Ø PRINT"< ... NOW SORTING ...>";
35Ø GOSUB 44Ø : REM REMOVE TO TIME
36Ø X=D:N=N+D:Q=Ø
37Ø IFA$(X)<=A$(X+D)THEN39Ø
38Ø Q=Q+D:Z=VARPTR(A$(X)):K=PEEK(Z):
    L=PEEK(Z+D):M=PEEK(Z+E):Y=VARPTR
    (A$(X+D)):POKEZ,PEEK(Y):POKEZ+D,
    PEEK(Y+D):POKEZ+E,PEEK(Y+E):POKE
    Y,K:POKEY+D,L:POKEY+E,M
39Ø X=X+D:IFX>I-1THEN4ØØELSE37Ø
4ØØ IFQ=ØTHEN41ØELSEIFX>I-1THEN35Ø
    ELSE37Ø
41Ø GOSUB 44Ø
42Ø PRINT "<SORTED - TOUCH ENTER>";
43Ø A$=INKEY$ : IF A$=CHR$(13) THEN
    1Ø ELSE 43Ø
44Ø CLS : FOR X = 1 TO I
45Ø PRINT A$(X) : NEXT X : RETURN
```

Program Listing 2

```
-----------------------------------------------------
              * Babybug II for Model I/III *
          Program memory requirement:  848 bytes
           Running time requirement:  162 bytes
        Basic normal overhead requirement:  83 bytes
  Memory Size can be set to total memory minus 11ØØ bytes
-----------------------------------------------------
1 CLEAR4:DEFINTD-Y:DEFSNGQ:E=1:F=256:G=16:H=48:I=55:J=64
2 CLS:INPUT"ORG MSB";R$:GOSUB13:POKE16527,M:L=M:INPUT"ORG
   LSB";R$:GOSUB13:POKE16526,M:K=M:Q=K+F*L:IFQ>32767THEN
   Q=Q-F*F
3 PRINT"ENTER HEX BYTES (X = STOP S = SKIP)
4 FORY=ETO8:GOSUB19:FORW=ETOG
5 P=26Ø+3*W+(Y-E)*J:PRINT@P,CHR$(91);
6 R$="":INPUTR$:IFR$="X"THEN12ELSEIFR$="S"THEN9ELSEIFR$=""
   THEN7ELSEGOSUB13:POKEQ,M
7 Q=Q+E:PRINTCHR$(27)CHR$(3Ø);:PRINT@P-64,R$;:K=K+E:IFK<F
   THEN8ELSEK=Ø:L=L+E
8 NEXTW:PRINT:GOSUB2Ø:GOTO11
9 PRINTCHR$(27);:FORX=QTOQ+(G-W):K=K+E:IFK<FTHEN1ØELSEK=Ø:
   L=L+E
1Ø NEXTX:Q=Q+17-W:GOSUB21:GOSUB2Ø
11 NEXTY:CLS:PRINT:PRINT:PRINT:GOTO4
12 INPUT"1 = RUN PROG, 2 = RE-ENTER";S:IFS<>1THEN2ELSEPRINT
   USR(Ø):GOTO12
13 R=ASC(RIGHT$(R$,E)):S=ASC(LEFT$(R$,E))
14 IFR>47ANDR<58THENT=R-HELSEIFR>JANDR<71THENT=R-I
15 IFS>47ANDS<58THENU=G*(S-H)ELSEIFS>JANDS<71THENU=G*(S-I)
16 M=T+U:RETURN
17 N=V/F:O=V-N*F:N=O/G:D=O-N*G:IFN>9THENPRINTCHR$(I+N);:
   ELSEPRINTCHR$(H+N);
18 IFD>9THENPRINTCHR$(I+D);:RETURNELSEPRINTCHR$(H+D);:
   RETURN
19 GOSUB21:FORX=QTOQ+15:V=PEEK(X):GOSUB17:PRINT" ";:NEXT:
   PRINT:RETURN
2Ø PRINTCHR$(27)TAB(7);:FORX=Q-GTOQ-E:V=PEEK(X):GOSUB17:
   PRINT" ";:NEXT:PRINT:RETURN
21 V=L:GOSUB17:V=K:GOSUB17:PRINT" * ";:RETURN
```

Program Listing 3

However, these time-consuming conversions are easier for me because I think in hexadecimal; I can do it in my head, since I only need to know the one-byte hex-to-decimal conversions. With practice, these one-byte conversions are easier to remember than multiplying by 256. My decimal 233, 66 address mentioned earlier turns into E9 42 hex, which (because it's stored backwards in Z80 terms) becomes address 42E9. In the Color Computer, decimal 30, 0 becomes 1E00 hex, which is already in the proper order. And since the Model I's disk and Level III Basic packages all include the &H option for hex numbers, I need convert no further. Extended Color Basic also includes &H as standard equipment (as well as HEX$—read on).

Toil and Trouble

There is a reason I brought up all these addressing confusions: like switching the catalog labels under the pieces of art, it's faster to switch the VARPTR labels to variables rather than the variables themselves. Program Listing 2 shows two ways of doing a simple bubble sort on the Model I. The normal sort starts at line 140, evaluates each item in the list against the next item in the list, and switches them (line 220) if they are out of order. A flag is set (variable Q) any time items in the list are switched; when all have been switched, Q is zero, and a completion message is printed. This sort essentially moves all the artwork every time the sort is passed through.

The VARPTR sort begins at line 290, defines all variables (to set aside all necessary space in the VARPTR table), and starts in much the same way as the usual bubble sort: an item is evaluated against the next one in the list. Then things begin to change: Line 380 is packed tightly to speed up the operation. Z is set to VARPTR(A$(X)), the address at which information about the current item being sorted is stored. Variables K, L and M are given the information about the item under consideration—its length, and where it currently resides in memory. Y is then set to VARPTR(A$(X + 1)), the address at which information about the next item in the list is stored. A$(X) is POKEd with PEEK(Y), PEEK(Y + 1), and PEEK(Y + 2), meaning the information about A$(X + 1) is being POKEd into A$(X). Likewise, A$(X + 1) is POKEd with K, L and M, the storage information about A$(X). The computer can now be fooled into believing A$(X) is A$(X + 1) and vice-versa. This line has switched the labels, not the artwork.

Is it worth the trouble? That depends

on the length of the sort you are doing. The fewer the items in the sort, the less advantage the VARPTR method will have. Table 7 compares the two sorting techniques.

It's clear that when memory space is at a premium, as it probably would be with large amounts of data to be sorted, the VARPTR method has a remarkable advantage—sorting 50 four-character strings in less than half the time an ordinary sort would need. It can also avoid potentially disastrous out-of-string-space errors.

The cause of the delay during sorting, by the way, is something called "garbage collection." When small computers such as the TRS-80 manipulate strings (such as the command **Z$** = **A$(X)** in Listing 2), temporary string space is set up for **Z$**. Each time **Z$** is used this way again, *new* space is set up for it, and the old version of **Z$** is abandoned. It's still there in memory, but it's inaccessible and the memory space is lost (temporarily!). Why would anyone create a Basic that did this? The answer is really quite reasonable: Since your TRS-80 Basic doesn't require you to define how long any string is going to be before you use it, Basic can't know ahead of time. If it was forced to use the same location to store **Z$** every time, and **Z$** got longer, it just wouldn't fit. To avoid any such conflicts, every time a string is defined—even if it has the same variable name—new space is set aside for it.

However, Basic has to recover this dead space sometime, and that process is called garbage collection. When Cleared string space is used up, operation of your running program is suspended, and Basic picks its way through string storage memory, cleaning out the dead strings and moving the live strings into those areas. When there are hundreds of strings involved, Basic has to check each and every one against its entire VARPTR table to see if it's alive or dead. That process can take up to an hour in complicated programs.

With the VARPTR sort, there is no garbage collection during the sort, because no string equations are used—only numerical pointer switching! That accounts for the remarkably consistent time for sorting, irrespective of the length of the items being sorted. The normal sort, on the other hand, takes longer and longer as it is required to pick up its garbage more often.

In Listing 2 I've only cleared a small amount of space (in line 60); if the cleared space were larger, the normal program would beat the VARPTR sort. But don't forget that most sorting programs will use massive amounts of data, thereby limiting the amount of string space you can clear. If you have a sorting program that runs a long time, try re-writing it with the VARPTR sort. It might save you a couple of hours.

Number of Items	Length of Items	Normal Sort Time	VARPTR Sort Time
30	1	0:34	0:49
30	3	0:36	0:49
30	5	0:47	0:50
30	7	1:39	0:51
30	8	OS ERROR	0:53
30	9	OS ERROR	OS ERROR
50	1	1:55	2:46
50	3	2:53	2:50
50	4	5:30	2:51
50	5	OS ERROR	OS ERROR

Table 7

```
---------------------------------------------------------
     * Babybug for Color Computer, 4K Color Basic *
         Program memory requirement:  864 bytes
          Running time requirement:  225 bytes
      Basic normal overhead requirement:  108 bytes
     Clear memory to total available minus 1197 bytes
                    (CLEAR 8,2750)
---------------------------------------------------------
1 CLEAR8:E=1:F=256:G=16:H=48:I=55:J=64
2 CLS:INPUT"ORG MSB";R$:GOSUB13:L=M:INPUT"ORG LSB";R$:GOSUB
    13:K=M:Q=K+F*L:ZN=Q:PRINT"ENTER HEX BYTES.  X=STOP, S=
    SKIP
3 FORY=E TO8:GOSUB21:FORW=E TO8
4 P=125+3*W+(Y-E)*32:PRINT@P," ";
5 R$="":INPUTR$:IFR$="X"THEN12ELSEIFR$="S"THEN9ELSEIFR$=""
    THEN7
6 PRINT@P-25,R$;:GOSUB13:POKEQ,M
7 Q=Q+E:PRINT@Y*32+71,"";:K=K+E:IFK<F THEN8ELSEK=0:L=L+E
8 NEXTW:GOSUB22:GOTO11
9 PRINT@Y*32+71,"";:FORX=Q TOQ+(8-W):K=K+E:IFK<F THEN10ELSE
    K=0:L=L+E
10 NEXTX:Q=Q+9-W:GOSUB22
11 NEXTY:CLS:PRINT:PRINT:PRINT:GOTO3
12 INPUT"1 = RUN PROG, 2 = RE-ENTER2;Z:IFZ<>1THEN2ELSEEXEC
    ZN:CLS:GOTO12
13 R=ASC(RIGHT$(R$,E)):S=ASC(LEFT$(R$,E))
14 IFR>47ANDR<58THENT=R-H ELSEIFR>J ANDR<71THENT=R-I
15 IFS>47ANDS<58THENU=G*(S-H)ELSEIFS>J ANDS<71THENU=G*(S-I)
16 M=T+U:RETURN
17 A=INT(V/F):B=V-A*F:C=INT(B/G):D=B-C*G
18 IFC>9THENPRINTCHR$(I+C);ELSEPRINTCHR$(H+C);
19 IFD>9THENPRINTCHR$(I+D);:RETURNELSEPRINTCHR$(H+D);:RETURN
20 RETURN
21 GOSUB23:FORX=Q TOQ+7:V=PEEK(X):GOSUB17:PRINT" ";:NEXTX:
    PRINT:RETURN
22 PRINT@Y*32+71,"";:FORX=Q-8TOQ-E:V=PEEK(X):GOSUB17:PRINT
    " ";:NEXTX:PRINT:RETURN
23 V=L:GOSUB17:V=K:GOSUB17:PRINT" * ";:RETURN
```

Program Listing 4

At Last

The purpose of this month's column (several thousand words ago) was to present again the Babybug machine-language monitor. Here is a brief description of the major actions of each version of this program.

In Program Listing 3 (the Model I or III version), string space is reduced to a minimum, and integers and other values are defined. The input prompts for most-significant and least-significant address bytes are presented, and the input strings are converted to numbers and

POKEd into the machine-language USR entry point. Variable Q is set to the current address being inspected; if Q is greater than address 32767, Level II Basic requires that it be converted to a negative number.

The user is then given a short menu of options (to enter hexadecimal bytes, to stop entry, or to skip the current line). The target address and the memory contents of it and 15 successive locations are displayed in hex. An arrow prompt points to the first hex byte. The options are (enter) to move on to the next byte; S (enter) to redisplay the first line and display the next one; X (enter) to terminate entry and present a new prompt; or the entry of any hexadecimal byte from 00 to FF.

A total of eight lines (128 bytes) may be displayed at any time; advancing to the next memory location clears the screen and presents a new line of information.

The X command (terminate entry) presents a new prompt asking the user either to run the machine code which has just been entered, starting at the initial entry address (the response to the MSB/LSB prompt), or to re-enter at a new address and begin the process again.

Major subroutines include hexadecimal string to numeric conversion (lines 13–16), integer to two-character hex string conversion (lines 17–18), full line display (line 19), revised line display (line 20), and address display (line 21).

The programming for the Color Basic version (Program Listing 4) differs from the Model I or III Babybug in two ways: the lack of the DEFINT function requires that integers be taken during calculation (line 17), and the absence of a USR function demands that the EXEC routine be used instead (line 12). Minor differences include the absence of an upward carriage return, and a required space separating variable names from subsequent keywords (line 3, for example). The display is similar but not identical because of the 32-character line limit on the Color Computer; only eight bytes of hex data are displayed per line (64 per screen).

The Extended Color Basic version of Babybug (Program Listing 5) is similar but faster than the Color Basic program because of the excellent HEX$ command. The entire numeric-to-hex conversion process is handled by the command V$ = HEX$(V) in line 17. The only modification I've made is the addition of a leading zero in front of single-digit hex numbers. A DEFUSR function is used instead of EXEC.

```
Model I/III:   21 00 3C CD 2B 00 28 FB FE 0D C8 77 23 C3 03 70
Color/Ex Col:  8E 04 00 BD A1 71 81 0D 26 01 39 A7 80 7E 0E 03
```

Table 8

```
 ---------------------------------------------------------
 * Babybug for Color Computer, 16K Extended Color Basic *
         Program memory requirement:  804 bytes
          Running time requirement:  190 bytes
     Basic normal overhead requirement:  88 bytes
      Clear memory to total available minus 1082 bytes
             (PMODE0:PCLEAR1:CLEAR8,4200)
 ---------------------------------------------------------
1 CLEAR8:E=1:F=256:G=16:H=48:I=55:J=64
2 CLS:INPUT"ORG MSB";R$:GOSUB13:L=M:INPUT"ORG LSB";R$:GOSUB
    13:K=M:Q=K+F*L:DEFUSR0=Q:PRINT"ENTER HEX BYTES.  X=STOP
    S=SKIP
3 FORY=E TO8:GOSUB19:FORW=E TO8
4 P=125+3*W+(Y-E)*32:PRINT@P," ";
5 R$="":INPUTR$:IFR$="X"THEN12ELSEIFR$="S"THEN9ELSEIFR$=""
    THEN7
6 PRINT@P-25,R$;:GOSUB13:POKEQ,M
7 Q=Q+E:PRINT@Y*32+71,"";:K=K+E:IFK<F THEN8ELSEK=0:L=L+E
8 NEXTW:GOSUB20:GOTO11
9 PRINT@Y*32+71,"";:FORX=Q TOQ+(8-W):K=K+E:IFK<F THEN10ELSE
    K=0:L=L+E
10 NEXTX:Q=Q+9-W:GOSUB20
11 NEXTY:CLS:PRINT:PRINT:PRINT:GOTO3
12 INPUT"1 = RUN PROG, 2 = RE-ENTER";Z:IFZ<>1THEN2ELSEPRINT
    USR0(0):CLS:GOTO12
13 R=ASC(RIGHT$(R$,E)):S=ASC(LEFT$(R$,E))
14 IFR>47ANDR<58THENT=R-H ELSEIFR>J ANDR<71THENT=R-I
15 IFS>47ANDS<58THENU=G*(S-H)ELSEIFS>J ANDS<71THENU=G*(S-I)
16 M=T+U:RETURN
17 V$=HEX$(V):IFLEN(V$)=1THENPRINT"0"+V$;ELSEPRINTV$;
18 RETURN
19 GOSUB21:FORX=Q TOQ+7:V=PEEK(X):GOSUB17:PRINT" ";:NEXTX
    PRINT:RETURN
20 PRINT@Y*32+71,"";:FORX=Q-8TOQ-E:V=PEEK(X):GOSUB17:PRINT
    " ";:NEXTX:PRINT:RETURN
21 V=L:GOSUB17:V=K:GOSUB17:PRINT" * ";:RETURN
```

Program Listing 5

Sample Session

Enter Babybug in Basic, and be sure to save a copy to tape before you run it, as random entry of hex information into memory can crash everything. After saving and verifying a copy, run the program. The screen will clear and this prompt will be displayed: ORG MSB?__. It is asking for the most significant byte of the address you wish to inspect, a two-digit hexadecimal number. For this first session, the screen will be examined. Enter 3C (04 on the Color Computer). The next prompt will be: ORG LSB?__. This asks for the least significant byte of the address to be examined. Enter 00 (00 on the Color Computer). The short menu (Enter Hex Bytes. X = Stop, S = Skip) will be displayed, followed by the complete address (3C00 or 0400) and sixteen (eight) hexadecimal bytes of data.

The first three bytes will likely be 4F 52 47. No, this is not machine code yet. These are the ASCII values for the first three letters at the top left of the screen—ORG. Type: 41 (enter) 42 (enter) 43 (enter). The bytes displayed should change as you enter each one, and the characters at the top left corner of the screen will change to ABC. Continue to enter hex data from 00 to FF and observe the screen display change. As you reach the end of the line, a new line will be displayed with the continued screen information. When you have entered 128 bytes (64 on the color computer), the screen will clear and the hex display will be continued.

Now type X (enter). You will see the prompt 1 = RUN PROG, 2 = RE-ENTER. Type 2 (enter). The screen will clear again, and the entry (origin) prompt will reappear. Before continuing, make sure you have saved a copy of the program! Enter 00 (A0 in Color Basic, 80 in Extended Color Basic) to MSB and 00 to

✓See List of Advertisers on page 386

LSB. The screen will now display the opening hex bytes of Basic. This is machine language, and this is a machine language program that you are going to run. Type X (enter), and in response to the run/re-enter prompt, type 1 (enter). You will see your computer's sign on message—Memory Size? or MEM SIZE? for Model I; CASS? for Model III; Color Basic 1.0 or Extended Color Basic 1.0. Babybug has been crashed, and your computer looks like you just turned it on.

Now reload Babybug (you *did* save it, right?), and run it. Respond to MSB and LSB with 70 and 00 (0E 00 on the Color Computer). Enter the bytes in Table 8 one at a time.

Enter X. Now run this program by responding 1 to the prompt. Nothing? Begin to type. Letters are displayed on the screen, starting at the top left corner. But no, you are *not* in Basic. This is a machine-language program which (a) calls a Basic keyboard scan routine, (b) checks for the enter key—carriage return code 0D hex—returning to Basic if it finds it, and (c) puts the typed character in screen memory.

You'll notice some interesting things when you begin typing with this routine. There is no cursor following the letters, since this program doesn't invoke a cursor display routine. Also, things like shift and shift lock (shift-0) produce odd characters on the Color Computer, and the backspace displays an H on all machines.

Here's a look at the hex code and what it does. The Model I or III program loads register HL with screen address 3C00 (21 00 3C), calls the ROM keyboard input routine (CD 2B 00), loops back to the keyboard routine if the value returned from the keyboard is zero (28 FB), compares to see if it is a carriage return (FE 0D), returns to Basic if it is a carriage return (C8), otherwise puts the character on the screen (77), moves to the next screen position by incrementing HL (23), and loops back to do the whole thing over (C3 03 70).

The Color Computer program loads the X register with the screen address 0400 (8E 04 00), calls the Color ROM keyboard routine (BD A1 71), checks if the value is a carriage return (81 0D), jumps ahead one place if it isn't a carriage return (26 01), otherwise returns to Basic (39). When it jumps ahead with a character, it puts it in screen memory (A7 80), and loops back to do it again (7E 0E 03).

The Z80 and the 6809E are different processors with unique properties; yet it's revealing to notice how similar these processes are, and even to compare them instruction-for-instruction. Although the hex codes are different, many of the properties are the same.

You're tired of testing and want something substantial? That's up to you. I will be happy to publish interesting routines created in hex code, as well as some of your experiences in developing them in hex notation alone. (No cheating, assembly language users!).

Updates

Look for some minor corrections and suggestions to the high resolution graphics board for the Model I (The Detailer, July) in next month's issue.

A printed circuit board is available for the Micro Front Panel (May). Write to me at Roxbury, Vermont 05669. ■

80 ACCOUNTANT

by Michael Tannenbaum C.P.A.

"Of all software packages, a financial report preparation package can be most valuable. . ."

Of all microcomputer software packages available, a Financial Report preparation package can be the most valuable to an accountant. The use of a microcomputer system to prepare these reports can be an enormous time saver.

Without a microcomputer report preparation requires an enormous amount of effort. First, a hand written report must be prepared and reviewed by the audit partner. Then a copy must be typed, proofed and compared to the original. Finally the whole report must be subject to an independent review. If a last minute change is required, the whole process of proofing must be repeated. Needless to say, the opportunity for an embarrassing error to occur is quite high.

The alternative of using a microcomputer to minimize this risk has been the root of the many requests I have received for information about report preparation systems. However, all the reports I have seen were part of General Ledger packages and were limited to an income statement and balance sheet. Reports such as the Statement of Retained Earnings, Changes in Financial Position, Analysis of Changes in Working Capital and Cash Flow Statement were missing. These reports are required to conform to professional standards.

Since the report is often used by third parties the standards of disclosure are higher than reports prepared for internal use. As a result, the reporting system must accommodate data such as footnotes where explanations of complex transactions or accounting policies must be presented. Although summary data may be adequate for outsiders, many clients require detailed financial reports for their own use. Thus many accountants prepare, in addition to summary reports, detailed departmental and comparative reports. These supplementary reports are usually presented in an appendix to the main reports.

New General Ledger System

The new three-disk General Ledger system (#26-4601) has been designed to cope with the problems of the professional and yet be usable in a small business environment. The ledger is designed to integrate with the other three disk systems, Accounts Receivable (#26-4604), Accounts Payable (#26-4605), and Payroll (#26-4603). If these other systems are used, summary data can be transferred at the month's end without the need for manual journal entries. The system has the capacity to handle up to 400 general ledger accounts and 4300 year-to-date transactions.

It is supplied on seven disks, three of which contain programs, and four data files. Two of the data files contain a sample general ledger with all accounts and reports previously defined. These sample data files are intended to provide a practice set for the owner of the system to use and become acquainted with. They are quite useful. The procedures to define accounts and output reports are complicated and the examples of data files that already work are helpful for reference purposes.

The program disks are arranged so that file maintenance programs are contained on program disk two and data entry programs are contained on disk one. This provides a sound method for separating file maintenance and up-date functions for general ledger processing. If these functions are separated, proper internal control over the ledger should be established.

Once a decision is made to automate general ledger processing, serious thought should be given to the arrangement of the output reports and the content of the financial statements prior to setup. Once installed, changes will be quite painful. The system contains an enormous amount of data. Balances are retained for each prior account and budgeted data by financial period. Thus, changes will involve shuffling lots of data.

File Setup

After the content and presentation of the financial reports has been determined, preparation and input of master file data can proceed. To aid in this process, the accompanying documentation devotes a good deal of space to file setup. I recommend that you read it several times before attempting the setup procedure. As with the accounts receivable system, an error in selecting an option in the initial files can cause a problem later on. The system uses decisions made when setting up the initial files to guide the setup procedure on the account and report files. For example, if a negative answer is made to the G/L cost center question on the company file maintenance screen, departmental reporting is inhibited and only a four digit account number is permitted.

Written in Cobol

The system is written in Cobol and like the other Cobol systems does not have full cursor controlled data entry screens. If an error is noticed in a data line after Enter has been pressed, the entire screen must be completed before corrections can be made. This feature and the strange assignment of special function keys will take some time to become familiar with. the Tab, ESC and F1 and F2 are used as special function keys. Tab is used to end a program and ESC is used to restart data entry in a field. F1 and F2 may have different functions depending on the program in use. Unfortunately not all of the screens indicate the operation of these special keys so you just have to remember what does what and when. While this is inconvenient, the other benefits of the system far outweigh this difficulty and all of the other three disk systems use the same conventions.

Report Generator

The key factor that distinguishes this system from all others that I have seen is the report generator. The report generator accommodates five types of statement formats and allows a great deal of customization. The reporting format types include the balance sheet, profit and loss statement, cash flow, statement of changes in financial position and analysis of changes in working capital. The different format types limit the column headings and ratio analysis options available but otherwise allow freedom over account combinations and presentations.

The means of implementing each format is found in the financial statement

layout procedures. The layout is accomplished by creating a data file with a description of the form. To provide instructions to the printing program a special syntax is employed. The instructions are arranged in sequential order and are executed in the same manner as a Basic program. Although the report specification procedures are complicated, the sample report specifications included in the sample data files can be referenced as a guide. By comparing these specifications with the printed reports the use of each type of report syntax will become clear.

In concept and implementation, the report preparation procedures are similar to commercial general ledger systems. Vendors of these systems, which usually cost many times more than the Radio Shack system, train customers in use of report generators in special classes. Typically class time ranges from a week to several days.

Running the sample forms will give you some idea of the report preparation potential of this system. Nine sample formats are provided ranging from a balance sheet to profit and loss reports. These formats can be combined with the data in a variety of ways. For example, a balance sheet can be prepared with and without comparative information. The comparative data can either be a budget or the prior year's end of month balances. The same format can be used for either type of comparative data. As a result more than nine output reports can be generated with the nine formats.

When printing these reports the system allows printing in any sequence desired. The sequence must be previously defined. This has an important advantage. If a summary set of reports and supporting schedules is required, then the whole set can be printed before any other printing is scheduled. Since the distribution of summary and detailed reports is usually quite different, this method of setting up a printing "job command language" for sequencing printed reports will be most appreciated.

Be prepared for a slow printing process. During the program evaluation process it took almost half an hour to print twelve reports with the limited sample data. While the computer is assembling the data, it appears to go to sleep. The screen indicates that processing is occurring and occasionally the drives are accessed but not much else is visible. The actual printing is also quite slow. The computer must do a great deal of processing since there is a rather long pause between lines.

However, the resulting reports are worth the wait. They reflect many thoughtful features which will be appreciated by the professional user. Account numbers are not printed. Dollar signs are inserted and foot notes are printed. A useful feature is the elimination of minus signs. In many reporting systems credits are identified by a minus or "cr" sign. In this system, if the balance is normally a credit or debit it is printed without a sign. If the balance is reversed, then the amount is printed with a parenthesis around it. This treatment conforms to normal practice for manual systems.

The system provides for the inclusion of text files. In the sample data only one text file is included—a disclaimer; however, up to 99 different files each with a maximum of 23 lines of textual data can be accommodated. This feature is ideal for an opinion, footnotes, segment reports and any other type of detail which must be appended to professionally prepared financial statements. Once a text file is created it is printed as specified by the statement layout description. There is no reason why text file two cannot be printed before text file one. This will be a big help to those who cannot decide the order of footnotes until the last possible moment.

Of course the source of all financial data for the reports is contained in the account files of the system. If departmental reporting is to be used these files will be quite extensive. When building the files, every bit of historical and budget information that will be required on the financial reports must be entered. This will be quite a considerable undertaking. There is a provision to enter budget and prior year month end balances for thirteen periods. Once all balances are entered, the final amount for the prior year's activity can be established as the opening balance for the current year's activity. This is done by running a special program appropriately called Setup.

Monthly activity is obtained from three sources: standard journal entries, general journal entries and other three-disk systems. The standard journal entries or recurring entries are sub-divided into two types, permanent and variable. Permanent entries are always the same amount and variable entries change each month. These entries, once defined, can be posted each month at the will of the system operator. If required, an automatic reversal entry can be specified. If that is done, the system will automatically generate the required reversing entries at the beginning of the next fiscal period.

Data entry from other three-disk integrated systems is accomplished by use of a transfer disk. Amounts to be transferred are extracted by a special interface program which offers the option of a transfer of details or the summary data for each account affected. We did not examine the transfer system during the evaluation process.

Data Entry

If the superior financial reports represent the good news, data entry can be considered the bad news. If data is to be entered through the general journal subsystem be prepared for slow data entry. Once the account number is entered, a two step procedure, the transaction date, amount, source, reference and reversal code must be entered. To reduce data entry time the system provides as default values prior data entered for the date, source and reference fields.

Obviously data for the system is best obtained through transfer from the other three disk systems. This condition should be easy to satisfy in a commercial environment where the other systems will be installed. However, in an environment where the accountant "writes up the books" on a monthly or annual basis a quicker method of data entry should be developed.

The slow speed of data entry is matched by the slow transfers from program to program. When a new option is selected on a menu, a caption appears on the screen—"Please Wait." After disk access, the selected program appears on the screen. If an error is made, a slow return to the main menu occurs and a slow selection of the new program is required. An examination of the disk directory provides a clue to this ponderous behavior. The system is segmented into many small programs which obviously take time to get in place to execute system functions.

This system, like the accounts receivable system, cries for a hard disk. If disk access times were quicker the system would be much more responsive. I suspect that when this system is installed in a more congenial environment than the floppy system it will behave with more alacrity. The floppy system also limits system capacity. In this system drive one is devoted to the five most frequently accessed files in the system. Only drive two is available for transaction data.

To increase the storage room available for transactions, it is possible to "compress" data already on file. To understand the process, consider that an account like wages could contain daily information from a variety of sources, weekly information and monthly information. Until instructed to "compress" the data, the system will retain all detail. The system offers the option of compressing data by date or

80 ACCOUNTANT

by period. If compression by date is selected, all activity for a day will be consolidated. Debits will be offset against credits and the net result reflected in the account. If the debits and credits offset, no entry will be reflected in the account. If compression is by period, only the net activity for the period will be reflected in the account.

Compression

Compression should be used with care. Once data is compressed a cross reference back to the original journal entry will be impossible. Probably a good method for retaining data for cross reference purposes will be to retain a backup prior to the compression procedure. Then, should a question occur, the cross reference run can be performed on the backup disk to locate the problem.

Documentation

Documentation for this system is provided in a large binder similar to the Accounts Receivable system. Index tabs divide each section. These tabs make using the documentation considerably easier than other Radio Shack offerings. Unfortunately the tabs have only paragraph numbers rather than descriptive names. This made referral to the index mandatory before a section could be selected. Hopefully this will be corrected in future releases. I found the documentation complete and informative. It included a section on general ledger concepts and contained two appendices on data files and handling inventory. With the documentation in hand, I found no difficulty in using the system.

In summary, if the account and transaction capacity do not disqualify this system, this package should be ideal for a practitioner with write up work. If the client can be induced to use the Model II for accounts payable and billing, data entry should be minimized. The General Ledger report generator will provide all of the required reports. Even if month end activity posted in the general journal is the sole source of data, the system will be useful. After all the financial reports are the professional's product and this system makes report preparation easy.

For the non-professional, the ability of the system to generate complex financial reports will be less important. However, because the system can generate departmental reports with comparatives and budgets, much valuable management information can be presented. In addition, the ability of the system to retain up to 4300 detailed transactions will be quite useful in tracking financial irregularities which could appear on the detailed reports.

This is indeed "big" system software at a micro price. I look forward to the next module in the "three disk" Cobol series. ■

80 REMARKS

Continues from page 8

our station is listening for any queries. When some amateur station comes on frequency and sends, say, an "M?" our station would automatically respond with a menu of the available bulletin board materials. The other station would then choose one item from the list and request it. . . "7." Our station would then supply the number seven item, which might be the latest in DX news, complete with a list of the stations active from rare countries, their frequencies, where to send confirmations of the contact, and so on.

Or we might send out a list of the latest FCC dockets affecting amateur radio or other recent FCC news. We could have a list of the traffic networks for handling messages, the recent repeater station changes, lists of other special interest nets, their times and frequencies, propagation predictions, a list of scheduled hamfests and conventions, new awards, and so on. There is a long list of information that hams might want to get and it would be available for the asking.

The plan will start out with 300 baud transmission, but soon revert to 1200 baud. We're going to encourage the experimentation with 9600 baud transmissions, since that would allow the transfer of information at around 7300 words per minute. At that speed we might be able to get most messages in between the dots of jamming CW stations!

The next step would be to develop automatic message relaying stations so that ham messages could be sent anywhere in the country in seconds. Imagine being able to drop a message to someone asking for a schedule in a few minutes on twenty meters! Where this would really come into its own would be during emergencies, when all the emergency traffic could be sent via automated means. Thus a ham could merely type the message on a small computer system, like the Radio Shack Pocket Computer, and it would be transferred to a central station for relaying to the addressee—much as Federal Express flies all packages to one city for sorting and reshipping the next day.

Emergency nets would then be set up with the control station polling each station in the net every few seconds, looking for messages. When the station had one, the polling would trigger sending the message. In this way no two stations would be transmitting at the same time. The message received, polling would continue.

We will need to develop error-correcting systems so that all messages are received perfectly. *80* is open for any ideas or articles along this line. ■

The Third World

A recent visit to the small Caribbean island of St. Lucia brought home to me the need for some sort of education for deserving people from the smaller countries of the world. The U.S. used to have the Point Four program, but that is long gone. Today, most of these students either have to make do with a poor education at home or else go to Russia or Cuba, where they are taught more politics than college.

Could we develop our proposed microcomputer school to the size where it could handle students from a hundred small countries? How would this be financed? Our government is trying to do all it can to stop spending, so they really don't need a new way to invest, even though it would be about the least expensive investment they could make, in the long run. These students are willing to work hard, not only at school, but also at part time work to get an education. I'll be visiting some more small countries in the next couple of weeks and will see what I can find in the way of answers to this problem. I'll be stopping off in Rio to set up plans for a South American edition of *Microcomputing*. Then Sherry and I will be off to South Africa to attend a microcomputer exposition in Johannesberg, where I'm a speaker. We'll be visiting Swaziland and Lesotho, two very small nearby countries, where ham licenses and stations for me to operate have been arranged. I'll be talking with the local hams and looking for ideas for practical ways to get Third World students the education they need. Indeed, their countries have a desperate need for skilled electronics and communications people.

Meanwhile, I will be dreaming about a college campus in New Hampshire, with modern buildings, heated mostly by solar heat, and surrounded by a couple hundred entrepreneurs and their growing electronics and computer businesses. ■

"You can try to design algorithms to emulate human behavior, or you can try to explain human behavior in terms of machines."

At SI, Machines Imitate Humans

The phrase 'artificial intelligence' summons up science fiction images of super-giant mainframe computers achieving sentience by sheer power and leading a revolt of the Moon colony against the government of Earth.

The truth is actually much stranger. At the Artificial Intelligence Laboratory at SI International (formerly Stanford Research Institute), Stanford, CA, one of the world leaders in this research, no huge light-blinking machine is achieving sentience. Rather, researchers use Digital Equipment Co. (DEC) PDP-10 computers programmed in Lisp to model aspects of human behavior.

Chuck Untulis, assistant director of SI International, stated that the PDP series is the standard machine in artificial intelligence research because it is designed for interactive work. "The IBMs are all batch process," he explained, "and the super-giants are just big scientific number crunchers."

SI approaches artificial intelligence from the computational or computer side. "You can try to design algorithms to emulate human behavior, or you can try to explain human behavior in terms of machines. We take a look at the way people behave in the world and try to emulate that behavior."

Although they do this entirely with software, they would like to have some specialized hardware.

"We keep talking to anyone who will listen about what we want in hardware," he stated. "Unfortunately, we're a pretty small organization, and none of the big companies listen, so we end up buying from the small firms."

Hard to Define

Untulis sees artificial intelligence research as going in several directions, but finds it very hard to define goals or even talk about the research because the concepts are so new the language has not adapted to it yet.

"We really don't know what artificial intelligence is, so there's no way we can define what our goal is. In the more traditional areas of science you can define your goal, and you know the methods you will use because they are all established. We work with concepts nobody has used before, and we're not sure how to talk about them, much less how to define our goals."

Many labs are heavily researching distributed intelligence. This concept postulates a large number of intelligent machines in independent operation in various parts of the world. Because of distance, different design and different use, they would have different information. One machine might have more up-to-date knowledge on a given subject than another.

Students Learn Psychology From Artificial Intelligence

While most people are interested in Artificial Intelligence for its practical applications in the computer field, at least one other use has been found. Dr. Homer "Tony" Stavely, a psychology professor at Keene State College in Keene, NH, is teaching a course using AI as a tool to illustrate some principles of psychology.

"Computers and Thought," as it is titled, will emphasize the "use of computers to simulate psychological processes," according to the course catalog. The course will also review early AI research as well as theory. Students will discuss the current state of the art and the possible future of AI. Textbooks will include *Artificial Intelligence* by Patrick Henry Winston and *Machines Who Think* by Pamela McCorduk.

Stavely, who has had an active interest in computers for several years, believes that there is a similarity between the thinking process of a computer. Though he concedes there are some obvious great differences, he said, "in the fundamentals of information processing, it doesn't matter if it is a machine or an organism... (there are) probably similar principles."

Stavely makes a convincing argument for using AI in teaching psychology. "The study of psychology is the behavioral relationship between organisms and their enviroment," he said. This "transactional relationship" produces behavior guided by the intake of information. "AI attempts to program machines to process information in ways which approximate the sophistication of (human) information processing," Stavely said.

There are three goals that Stavely has set up for his students to achieve. He wants to increase their understanding of the nature of perception and thought; to demonstrate that computers are tools to aid in their thinking; and to show that computers are inportant as a simulation device. This last goal emphasizes the use of a computer to set up a hypothetical situation to "see what happens if..."

Stavely sees AI programming giving computers some ability to perceive problems, organize and remember information and, ultimately, write programs on their own to control their databases.

The future of AI is promising, Stavely believes, especially in the educational field. He said that hardware capabilities far outstrip the capabilities of the software that has been produced. He thinks that AI may be the answer to this software problem. "AI is the area in which we are learning the software skills to bring about advances in using computers as tools to think with," he said. ■

by Michael Nadeau
80 Microcomputing *staff*

✓See List of Advertisers on page 386

If, then, one machine asks another a question and receives data back in answer, how does the first machine handle the data it receives?

"You get into the whole question of belief structures. Each machine has a belief of what the world is, and that belief might be inaccurate." The question is how these belief structures are modeled, limited and changed.

Expert Systems

Another area of artificial intelligence is rule, or expert, systems, in which researchers preserve and communicate to others the special understanding of an expert (an Einstein, for instance).

The hope is to capture the expert's world view in a finite number of data statements to quantify it. "You do that with a set of rules which set up a model which can be given data or asked questions."

Another area involves designing programs that will handle normal language, either in the form of keyboard input or, further down the road, verbal input. The goal is to eliminate the need for programming languages to communicate with the computer. The businessman could talk with his machine just as he can with a human employee; if he needs to know how many widgets he has in St. Louis, all he has to do is ask. The computer will decide what data bank to query and exactly what piece of information is wanted.

Vision Input

Vision research is another area in which SI International is active. Humans have a very high talent for interpreting visual data; it is difficult to approximate that ability with a machine. SI is developing programs which allow the computer to digitalize a picture and not only understand the literal data but draw conclusions from clues.

For instance, they would like the machine to locate a hidden light source from such clues as the size and placement of shadows. It then should be able to describe the expected reflectivity of objects from the inferred position of the light source.

One goal of artificial research is the self-programming computer, the machine which can decide for itself how to go about doing something and change its ideas on the basis of experience. According to Untulis, SI is working on the theoretical end of this question by designing theorem-provers. These are methods whereby a machine, given an idea or theory, can experiment to determine whether the theory is correct. If it is not, the machine will modify it to fit the experimental data, much as a scientist works in a lab.

A Question of Difference

All of this involves one of the basic questions of artificial intelligence: Is it qualitatively different from human intelligence or does it only appear to be different at this stage because it is so primitive?

Computers are famous for their ability to instantly memorize and manipulate huge lists of items. Humans cannot do this nearly as quickly or as well. Yet, machines cannot make decisions on their own—humans have to tell them precisely what should be done with their data.

The answer, Untulis explained, "depends on whether you are talking on the short run or the long run. If you are talking on the short run, then it is both qualitatively and quantitatively different. If you are talking about the long run, then it is not." However, the long run probably means anywhere from 200 to 1,000 years.

Researchers are presently working with limited domains, try to construct algorithms that emulate human behavior, while dealing with general problems in a theoretical manner.

So why is this in a magazine on microcomputers? What relevance does it have for the TRS-80 user?

Cognitive Science

The research into artificial intelligence falls under cognitive science, a new branch of science that includes aspects of both computer science and psychology. The fallout from the research in this area is likely to influence the design of the machines we will be using as well as our concepts of ourselves in the 1990s and beyond.

Untulis mentioned he sees a couple of areas in which artificial intelligence research may soon be influencing software design, although he cautioned that the issues are far from clear.

One area in which machines can do a better job than humans is very large-scale integration in, for instance, circuit design. One company is trying to adapt planning and deduction methods worked out at SI International for multiple-level planning to this sort of operation.

In multiple-level planning the machine works on an abstraction ladder. It starts at the top, forming a very generalized plan, or, in this case, circuit design concept. It then moves down a step and makes a more particularized design concept based on its generalized work. In this way it works from step to step towards the actual design of the circuit.

> *"The hope is to capture the expert's world view in a finite number of data statements. . ."*

If its tests show that the most recent plan is faulty, it moves back up the ladder to more generalized plans and refines its steps.

Visual interpretation would also be very handy for a machine that designs or tests circuits using this multiple planning method.

"All these things go hand-in-hand," Untulis told us. "We want the machine to keep track of the errors we make in designing one circuit, for instance, so it will be able to warn us if we start to repeat them in the next circuit design."

These ideas may be used in the future but they will not be here tomorrow.

"The things we're talking about are 10 to 20 years from fruition. We're just thinking about these things today." ■

Radio Shack, Ohio Publisher Plan Joint School Market Assault

Tandy/Radio Shack, Fort Worth, TX, and South-Western Publishing Co., Cincinnati, OH, a leading business education and business administration publisher, will work together to develop and market educational courseware for TRS-80 microcomputers.

According to William D. Gattis, director of Radio Shack's Education Division, South-Western will help Tandy in the development and marketing of programs for Models I and III, and Radio Shack, in return, will provide computers for demonstrations and displays by South-Western at major educational shows and conventions.

Radio Shack will also develop special software protection features for the Model I similar to those already available for the Model III, to prevent unauthorized copying of South-Western's software.

South-Western is a subsidiary of SFN, Chicago, IL, a large elementary and high school education publisher. ■

Microcomputers Enter Medicine

Computers have been used by the medical community for nearly two decades. In May of 1980, *Modern Healthcare* reported that community hospitals spent more than 1.25 billion dollars a year on computerized business and medical information systems. The *Journal Of The American Medical Association* found, in its report on "Computers in Medicine" (Dec. 1978), that computers were being used routinely in hospital laboratories, radiology departments, emergency rooms and pharmacies. Not only were computers maintaining patient charts and billing records, they were also capable of such high level tasks as predicting the probability of gangrene infections in appendicitis victims.

Since their widespread introduction four years ago, microcomputers have also begun to have an impact on the practice of medicine. The vast majority of physicians who use microcomputers are primarily interested in office management, but according to Gene Thompson of Thompson Consultants, Seattle, WA, "Thirty percent of the doctors who have been exposed to microcomputers are learning to program."

This is because doctors appreciate the potential of the microcomputer and realize they themselves "will create the applications—once they learn to program," says Thompson.

Hospitals are using microcomputers in increasing numbers to handle specialized tasks. Many such tasks could be performed on hospital mainframe computers but this creates a number of problems—timesharing limits accessibility, new software is extremely costly for mainframe computers and set-up procedures are costly. With a microcomputer in the lab to calculate blood gases or in the pharmacy to assist in the preparation of prescriptions, hospitals can greatly improve efficiency without a large financial investment.

"One hospital I know of uses a microcomputer to calculate hormone balances, others are using them to do small office tasks, and one pathologist uses a text editing program and a micro to keep notes on patients," said Thompson. "Micros are especially useful for recording and tabulating laboratory protocols, which are the results of visual tests and inspections in fields like bacteriology and microbiology." Thompson speculates that microcomputer use in medicine will continue to grow and that software exchange among physicians will become a common practice.

There is a precedent for this kind of software exchange. Robert Kinch of the Forsyth Memorial Hospital in Winston-Salem, NC, is the President of ECHO, an IBM users group whose members are hospitals and doctors. This group exchanges software across the entire spectrum of medical related applications, including ambulatory care, clinical management, census management and laboratory/pharmacy use. There is a similar group for microcomputer users as well as several medically-oriented try. These bulletin boards will likely play a key role in the development of physician's key role in the development of physicians user's groups.

Commercially produced software is also available. Radio Shack sells a Medical Office System package for use with the TRS-80 Models I and III. Dr. Larry Stoneburner runs MedLogic Systems, a Los Angeles, CA, firm selling software for both administrative and medical applications.

Thompson, however, thinks most doctors are "skeptical about prepackaged software." Hence, many physicians are becoming programmers.

One example of the way a microcomputer can make a hospital more efficient is the General Hospital in Ventura, Calif. As reported in *The Journal Of Family Practice* (Vol. 11, No. 4), the hospital's resident doctors were tying up phone lines to the outpatient clinic with constant checks to see if their patients had arrived, when their next appointment was scheduled and the like. With 35 physicians using the clinic, the load became hard to manage. The solution proved to be a microcomputer.

The hospital had a closed-circuit television system already in place. It was a simple matter to patch the microcomputer into that closed circuit system. The outpatient schedule could then be entered into the computer by a secretary and displayed throughout the hospital on a regular basis. The software was designed to permit not only a display of the clinic's schedule, but also to indicate whether a patient had arrived for his scheduled appointment.

The physicians were able to conduct their rounds in the hospital while periodically consulting the televised outpatient schedule to find out when they needed to be in the clinic to see a patient. This system cut down on the volume of doctor's telephone calls to the clinic by 40 percent in the first four weeks of operation. The physicians involved overwhelmingly approved of the system.

Still, while microcomputers are catching on with the medical community, some doctors and administrators harbor reservations about system capability. The Canadian Medical Association's Executive Secretary, writing in the March 22, 1980 issue of the CMA Journal, stated flatly that microcomputers were too small and too unreliable for use in medical practice. B. E. Freamo cautioned physicians to wait to see if good software was developed and to demand that hardware vendors provide service contracts as a condition of sale.

In spite of these kinds of cautions and what Gene Thompson calls the "innate conservatism of physicians," don't be surprised if you see a TRS-80 in your doctor's office sometime in the near future. ■

by G. Michael Vose
80 Microcomputing *Staff*

Medical Journals Cover Computers

As if doctors didn't already have enough literature to keep up with, there are now several publications circulating information about clinical uses of microcomputers. There is a nationwide, general purpose publication called *National Report: Computers And Health;* there are general information exchange newsletters such as *Dr. Com Puter's Report* and the *Apple Mug Newsletter;* and various specialty newsletters for dentists and psychologists.

Most of these publications are concerned with one thing—finding out who's doing what and how to contact them. The name of the game is information exchange. There is apparently no greater consumer of information than medical doctors.

In February Dr. Larry Stoneburner of Orange, CA started a newsletter as a method of information exchange among the Apple Medical User's Group. Called the *Apple Mug Newsletter,* the publication has grown from five subscribers to its present 400, in just five months. *Apple Mug* is a member of the *International Apple Core,* an international organiza-

tion of Apple Computer Clubs.

The *Apple Mug Newsletter* features news about doctors using computers in their practice, particularly clinical applications. It is a forum for the exchange of ideas among its members. Although the *Apple Mug* is primarily composed of Apple computer uses, the *Newsletter* is interested in developing a rapport with CP/M users in order to obtain a broad base of software techniques, ideas and applications.

The *Apple Mug Newsletter* has plans to present a Medical Applications Seminar at the West Coast Computer Faire in San Francisco next March, and is currently soliciting papers for presentation during this seminar.

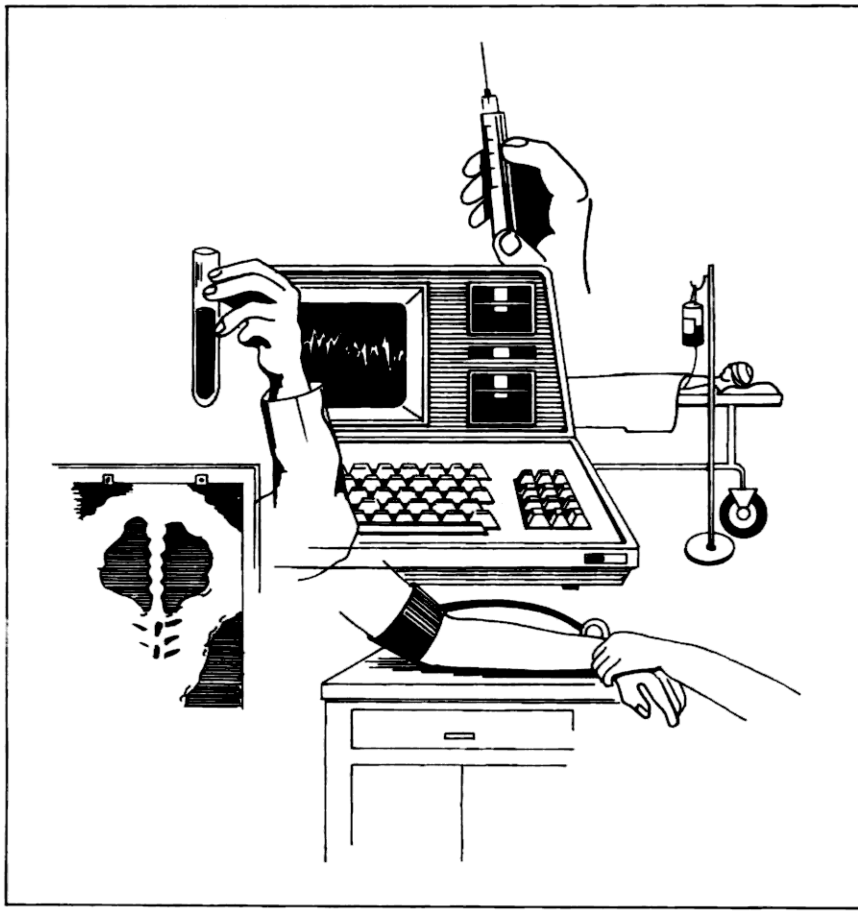
On the east coast, there are two medical microcomputer publications coming from the same office. The *Medical Computer Journal* and *Dr. Com Puter's Report* are published quarterly by Dr. Azis Ghaussy of East Hampton, CT. Dr. Ghaussy's journals not only provide a forum for the exchange of news and ideas—they also publish medical application program listings. With well over 1000 subscribers, including 75–100 international subscribers, the *Journal* and the *Report* reach a wide variety of physicians using microcomputers. Both publications print software listings for Apple, TRS-80 and North Star Horizon computers.

Dr. Ghaussy's journals not only provide a forum for the exchange of news and ideas—they also publish medical application program listings.

The *Journal* and the *Report* seldom pay for submissions; physicians share software in an effort to help their patients by keeping the information exchange dynamic. ■

The publications mentioned in this article may be contacted at the following addresses: National Report: Computers and Health, *PO Box 40838, Washington, DC 20016;* Apple Mug Newsletter, *2914 Katella Suite 208, Orange, CA 92667;* Medical Computer Journal *and* Dr. Com Puter's Report, *Dr. Aziz Ghaussy, 42 East High Street, East Hampton, CT 06424.*

by G. Michael Vose
80 Microcomputing *staff*



Medical Software Exchange Helps Docs Sell Programs

As physicians begin to develop an appreciation for the potential of the microcomputer, they apply their medical knowledge to the task of producing software with medical applications. Many physicians try to market the result of their labors but find it extremely difficult to track down other computer doctors. Enter SoftDoc, a microcomputer software exchange.

SoftDoc is the creation of Dr. James Gange of Los Angeles, CA. Dr. Gange began SoftDoc as a way to help some like-minded physician friends who all had some software they were willing and eager to share with their colleagues. Since its inception about a year ago, SoftDoc has been adopted as an official project by the Society for Computer Medicine, whose President is Dr. Marion Ball, the Director of Computer Services at the Temple University Medical School.

SoftDoc's current offering is a CP/M based collection of approximately 25 programs. Included in the package, distributed on an 8-inch floppy disk, are programs to:

- evaluate pulmonary function;
- set up a respirator;
- generate evaluations of lab tests such as glucose tolerance and electrolyte balance;
- evaluate a patient's lifestyle, help the patient figure out why he cannot give up smoking, and suggest ways to improve health by changing lifestyle habits;
- the ANSI standard Mumps program, the most commonly used medical usage language.

All these programs are written in various dialects of Basic (with the exception of Mumps). The disk is available to physicians for a price, or a physician can join SoftDoc and receive the software free by contributing software to the exchange.

The program will run, with some

modification, on the TRS-80 Model II under the CP/M operating system.

SoftDoc is always on the lookout for new software and is looking for competent software editors and programmers, as well. Dr. Gange says the exchange cannot pay editors at present but can arrange to provide references in return for editing, translating to other Basics and operating systems and debugging. SoftDoc is not interested in business office software; the thrust of SoftDoc is clinical application software.

In a recent interview with *80 Microcomputing*, Dr. Gange said that, theoretically, microcomputers should have a profound impact on the practice of medicine. Practically, however, he said, "this hasn't happened yet. The development of clinical applications is still in the embryonic stage."

Dr. Gange speculated that growth in the field had been slowed by the difficulty of distributing medical software. This was one of the reasons he has launched SoftDoc.■

For further information, write to:

SoftDoc
c/o Dr. James Gange
1433 Roszmare Road
Los Angeles, CA 90024

by G. Michael Vose
80 Microcomputing *staff*

Computer Aids Help Handicapped

Computer aids now make it easier for the handicapped to work, communicate with others and seek emergency help. They assist the hearing and visually impaired as well as those severely handicapped from cerebral palsy or spinal disorders.

The most widely used device is the TTY/TTD (teletypewriter telecommunications device), which allows the deaf to communicate over telephone lines. The National Crisis Center for the Deaf has a 24-hour hotline that receives TTY/TTD communication concerning medical and emotional problems and dispatches emergency help to callers. "They receive 10 to 15 calls a week," says director Mary Compton.

A prototype conversion of the TTD baudot to ASCII code has been developed by Paul Rinaldo of the Amateur Radio Research and Development Corporation (AMRAD), using the TRS-80 Model I. Rinaldo has also made prototypes using the Commodore PET and Apple computers. This would allow these machines to replace the teletype terminals as home communication devices.

Two states offer TTD/TTYs at low or no cost. Michigan Bell now sells the device after a Public Act mandated they be made available at cost of production. Since March 1981 they have installed 800 TTDs.

In California, phone companies are required to supply the devices at no cost; they have until 1984 to comply with the legislation. An original bill required TTDs be available at $14.50 a month, and they were installed in the pilot program in Fremont, CA.

Microprocessor

Most TTDs feature a single chip microprocessor that converts baudot to ASCII. Printer and display are either separate or one unit.

A number of companies are producing the TTDs, including Plantronics Inc, whose VuPhone has both baudot and ASCII; Applied Communications Corporation, whose Phonetype was the first TTD developed; Specialized Systems Incorporated, whose customers include Sears Roebuck, Bank of America and 32 state governments; and Krown Research, whose Porta Printer Plus offers ASCII code format for $100 over the $575 price tag. Various teletypewriters are also available.

Automated Data Systems of Madison, WI, produces an alternative to the TTDs called Superphone, a terminal with voice synthesis that can be used with any touchtone phone. It allows direct contact that the TTY doesn't, since the receiver of the call doesn't need a TTY terminal. The Superphone terminal also can be attached to a television set to produce a CRT screen and has features such as an automatic answering device.

For face-to-face communication, the deaf and dumb can use Automated Data's VIP communicator, a pocket-sized teletype with an attachment worn on the person's lapel flashing the letters being typed.

For the Blind

Terminals are also being modified for the blind. A Low Vision Terminal (LVTS) prototype has been developed by M. Daniel Simkovitz, an engineer at Wayne State University in Detroit. The screen shows characters three inches high and allows the partially sighted to read serially or a few characters at a time.

"The partially sighted are not understood," says Simkovitz.

Most work has gone into synthetic voice for talking computers or braille keys with a voice output. But 80 percent of the 347,000 legally blind still have some vision, and many of those could use the LVTS, he says.

There are also printers available for the partially sighted. Automated Data offers printers and CRTs with large characters and a computer with braille printer output.

Voice synthesizers use either electronic phoneme synthesis or the more subtle allophone synthesis. Most micros, including the TRS-80, have single chip voice synthesizers available.

But not all computer commands can be represented by speech, Simkovitz says, which makes the LVTS more valuable to the partially sighted. Besides, the partially sighted often don't read braille.

Interfaces are being produced that allow the severely handicapped to operate a keyboard. The Express I & II from Prentke Romich Company, Shreve, OH, are key interfaces with serial ASCII data that converts to parallel ASCII. The interface features a panel of 120 blinking lights. A joystick or optical head pointer is used in place of the keyboard. When the user makes the selection by an optical head set, a microprocessor receives the signals via a small tube with a lens and light sensor.

These devices should help the nation's 21 million disabled in their homes and jobs. Those developing computer aids hope to increase job opportunities for the handicapped.

Superphone "could make a difference, especially when employing the deaf," says Rob Engelke of Automated Data. "Even a stock clerk . . . has got to be able to contact people."

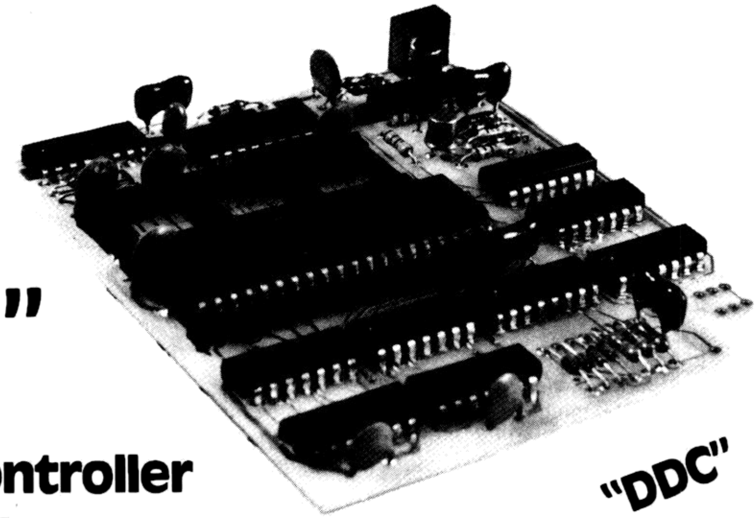
Electronic Silly Putty

A writer in the Midwest is using the Express I with printer hookup to produce a monthly column. Engelke says he's had some orders from businesses, and computer companies are snapping up his terminals because they can be modified for other applications.

"It has a bus, it's just like an Apple," he says. "There's plenty of room in there. It's electronic silly putty in a keyboard."

Institutions have also been buying products for the handicapped. Now that

Aerocomp's Proven Best-By Test! The "DDC"



Double Density Controller

★ Technical Superiority

At last! A double density controller for Model I with HIGHER PROBABILITY OF DATA RECOVERY THAN WITH ANY OTHER DOUBLE DENSITY CONTROLLER ON THE MARKET TODAY! The "DDC" from Aerocomp. No need to worry about the problems that keep cropping up on existing products. AEROCOMP'S new analog design phase lock loop data separator has a wider capture window than the digital types currently on the market. This allows high resolution data centering. The finest resolution available with digital circuitry is 125 ns (nano seconds). The "DDC" analog circuit allows infinately variable tuning. Attack and settling times are optimum for 5-1/4 inch diskettes.
The units presently on the market use a write precompensation circuit that is very "sloppy". Board to board tolerance is extremely wide - in the order of ± 100 ns. The "DDC" is accurate to within ± 20 ns.
The botton line is state of the art reliability!

★ Test Proven

Tests were conducted on AEROCOMP'S "DDC", Percom's "Doubler A"* and "Doubler II"* and LNW's "LNDoubler"** using a Radio Shack TRS80*** Model I, Level 2, 48 K with TRS80 Expansion Interface and a Percom TFD100* disk drive (Siemens Model 82). Diskette was Memorex 3401. The test diskette chosen was a well used piece of media to determine performance under adverse conditions. The various double density adapters were installed sequentially in the expansion interface.
The test consisted of formatting 40 tracks on the diskette and writing a 6DB6 data pattern on all tracks. The 6DB6 pattern was chosen because it is recommended as a "worst case" test by manufacturers of drives and diskettes. An attempt was then made to read each sector on the disk once - no retrys. Operating system was Newdos/80, Version 1.0, with Double Zap, Version 2.0. Unreadable sectors were totalled and recorded. The test was run ten times with each double density controller and the data averaged. Test results are shown in the table.

★ Features

TRS80 Model I owners who are ready for reliable double density operation will get (1) 80% more storage per diskette, (2) single and double density data separation with far fewer disk I/O errors, (3) single density compatibility and (4) simple plug-in installation. Compatible with all existing double density software.

★ Value

$149.95 for the BEST double density controller on the market.
$189.95 for "DDC" complete with DOSPLUS 3.3D
$239.95 for "DDC" complete with LDOS

★ TEST RESULTS ★

MFR & PRODUCT	SECTORS LOCKED OUT (AVG)
AEROCOMP "DDC"	0
PERCOM "DOUBLER II"	18
PERCOM "DOUBLER A"	250
LNW "LNDOUBLER"	202

Note: test results available upon written request. All tests conducted prior to 8-25-81

Aerocomp's 14 day money back guarantee applies to hardware only.
Specials will be prorated. Shipping $2.00 in Cont. US. See opposite page for details.

THE FIRST PRODUCTION RUN WILL GO FAST - ORDER TODAY!

Data Separators

The advances that make the "DDC" great are incorporated in the new AEROCOMP Single Density Data Separator ("SDS") and Double Density Data Separator ("DDS").

★ Has your original manufacturer left you holding the bag?

If you already own a Percom "Doubler A", "Doubler II" or LNW "LNDoubler", the AEROCOMP "DDS" will make it right. Look at the test results:

MFR. & PRODUCT	SECTORS LOCKED OUT WITHOUT "DDS"	SECTORS LOCKED OUT WITH "DDS"
PERCOM "DOUBLER II"	18	1
PERCOM "DOUBLER A"	250	0
LNW "LNDOUBLER"	202	0

Note: Same test procedures as "DDC".
* Trademark of Percom Data Co.
** Trademark of LNW
*** Trademark of Tandy Corporation

★ "DDS"..... $49.95
(Use 1791 chip from your DD Controller)

★ "DDS" with disk controller chip included....... $79.95

★ Disk controller chip..... $34.95

(Shipping $2.00 Cont. US - see opposite page for details)

Plugs directly into your existing Double Density Controller.

Do you need a Single Density Data Separator?

The internal data separator in the WD1771 chip (R/S Expansion Interface) is NOT recommended by WD for reliable data transfer. Do you have any of these problems: Lost data, tracks locked out, CRC errors, disk retry? YOU NEED ONE!

★ "SDS"...... $29.95
(For Mod. I; shipping $2.00)

See opposite page▸▸▸▸▸▸

✓See List of Advertisers on page 386

Michigan Bell has made the TTD available, several offices and stores have purchased the equipment for employees.

"The cutting edge for this type of device is employment," says Simkovitz of his LVTS. Many partially sighted people "have steered away from some professions because of obstacles they face with computer devices."

Simkovitz, who is himself legally blind, says "For many years I faced frustration with computers." He's hopeful others will develop further applications of the device.

There is a program currently placing handicapped people in the job market. National Quest, a part of Unicos Corporation, Leominster, MA, will train the handicapped in computer-related fields and place them in industry. Integral to the program is the talking computer Unicos has developed that reads back input and output to the blind. The handicapped could also work at home, sending reports to their employers through a modem.

Interest in computers for the handicapped is growing, and Johns Hopkins University hopes to harness some of this enthusiasm with a Personal Computing to Aid the Handicapped contest, sponsored jointly with the National Science Foundation and Radio Shack. Entries will be judged in three categories: computer-based devices, computer programs or system concept/design. The entries will be displayed in ten regional fairs with a final exhibit from Oct. 31 to Nov. 1 at the National Academy of Sciences, Washington, DC. Johns Hopkins will eventually have a two-day workshop to unite the winning inventors. They've received over 800 entries so far.

"For face-to-face communication, the deaf and dumb can use . . . a pocket-sized teletype with an attachment worn on the person's lapel flashing the letters being typed."

Market Problems

Though interest in computers is high, it's sometimes hard for inventors to find companies to market their products. Paul Rinaldo is still trying to find someone to market his ASCII prototypes for the TRS-80, Apple and PET. It's particularly hard to find a distributor for the TRS-80, since the Model I is out of existence. Creating a Model II or III prototype "would mean starting all over again," he says. The Model I prototype took two years to develop.

Now Rinaldo is planning an S100 interface and prototypes for the TI and Atari computers. He's received federal grants of $46,000 and $50,000 in the past two years.

Simkovitz's project was underwritten by the university "way beyond their initial commitment," he says. But budget cuts made further funding impossible, and he needs $30–50,000 to complete the LVTS so it's ready to market.

Support from the outside has been "shameful and shabby," he says. He's

talked to several companies, and half a dozen have seen the prototype, but he's had no offers yet. Simkovitz would like to take the product to a small company experienced with products for the partially sighted.

Costs

The cost of computer aids may be steep for the user as well. The LVTS should retail for between $2500 and $4500. The Express I interface is $2380; the Express II, with full ASCII character set but no printer, is $1900. Prentke Romich's environmental control units, which hook up to electrical appliances in the home, run from $560 to $1325. The hand-held teletype VIP communicator is $199 with an $89 voice attachment. Automated Data also offers a computer terminal for the blind with speech output for $900 to $1000.

The TTDs, in states that don't provide them through the phone company, run about $550. In Michigan, Bell Telephone provides a TTD for $400.86, or $6.68 a month for five years. You can lease the device for $125. In California, users pay only for regular telephone service.

More affordable are information services including the bulletin board service SpecialNet and Creative Computing for the Handicapped, a West German organization.

Though SpecialNet is directed at special education administrators, its bulletin boards will feature information useful to the handicapped. A litigation bulletin board will be managed by the editors of the *Education of the Handicapped Law Review;* the Prentke Romich Company will operate a bulletin board on system devices for the handicapped; and a federal bulletin board will contain pending legislation of importance to the handicapped, including federal budget actions. SpecialNet will also feature electronic mail.

> *"The cost of computer aids may be steep for the user as well."*

The Association for the Advancement of Microcomputer-Based Work at Home for the Handicapped, West Germany, will translate articles on microcomputers onto cassette for the blind. They also will translate advertisements with information on how to order software and hardware from American distributors.

Prentke Romich may organize a review board made up of special education teachers and the handicapped who would evaluate educational software. Their initial network project, Apple Computer Clearinghouse for the Handicapped, has been scaled down, though they may compile a service catalog of software for the handicapped, says computer manager Neil Russel.

Computer companies are offering more services and products for the handicapped, and finding a market for them. All they have to do," says Engelke, is "use their imagination."■

by Betty Thayer
80 Microcomputing *staff*

Computeens Mob Camp Keep Prof Jumping, Happy

For the fourth year in a row, Dr. Michael Zabinski (of Fairfield University, New Haven, CT) and a group of experienced teachers spent two weeks this summer with more than 100 teenage computer nuts.

Taking place during the last two weeks of July in the Klar Crest Resort in Moodus, CT, the fourth annual National Computer Camp was a complete success, according to Dr. Zabinski. He told us more than 160 campers aged 10 to 18 of both sexes came and had a good time.

The camp, the first of its kind, combines small group instruction in microcomputer uses with strong emphasis on the TRS-80 with normal camp activities. Students this year came from as far away as London, England, and Caracas, Venezuela.

One hundred ten students attend the camp for a week, and about half of them stay for the second week. The staff includes 188 teachers plus two recreational organizers, two directors, a medical doctor who is doing computer research and Dr. Zabinski.

The students work in small groups arranged according to their level of knowledge, ranging from beginners to two groups of Assembly programmers. All work on TRS-80s, except one of the Assembly programmer groups which used Apples and a Wang minicomputer.

A typical camp day starts with a morning computer session from 9 to 11 a.m. There is a midday break for sports, including swimming and diving in the camp pool, and soccer with a college varsity player who is a physical education major. After lunch the students relax until 1:15 when they go back to the computers until 5 p.m. At night Dr. Zabinski runs a movie or other special event. One night they conducted a Las Vegas night, with games of chance run on the microcomputers.

Most of the computer work is done in a spacious and well-lit camp hall. The campers worked in an informal atmosphere, moving around a lot and sharing information and activities.

"This is not a classroom setting," Dr. Zabinski explained. "We give them books but they are not textbooks and we never tell them to turn to page such and such. We use them for reference only."

The campers are often too dedicated to their computer work. In spite of the beautiful, quiet grounds of the resort camp and the bright weather, Zabinski had to personally order some of the kids out of the hall to take a swim at the 11 a.m. break.

Rick Larcom, a teacher from a New Canaan, CT, high school who runs the TRS-80 Assembly group, mentioned he has to turn out the lights at night to force the kids to leave and he has to lock the doors or he'll wake up to find some of them already at work early in the morning.

Dr. Zabinski feels that the secret to the success of his camp is the staff. "I handpick everybody. I train most of them myself. Most of my teaching assistants used to be campers. I emphasize that I have to have a good staff. They're knowledgeable and experienced at what they're doing."

The teachers come from all levels, from elementary through high school, and Dr. Zabinski tries to match the teachers with age groups they are used to. This isn't always easy or even possible—the groups are divided according to the students' level of accomplishment, not age, and especially in the upper levels the ages are fairly mixed.

The emphasis of the camp is not solely on learning hard technical material. In fact, Larcom teaches social studies, not math, during the year.

Zabinski isn't worried about technical orientation among his teachers so long as they know their subjects well and are good with the campers. His concern is having the students learn something.

His formula is proving very successful. "When the kids come back year after year and bring their friends, it tells you something about how well you're doing," explained Dr. Zabinski, who added the camp does practically no advertising because it fills up so fast from repeat campers and people who have heard of it by word of mouth.

Next year they are considering expanding to four weeks and may run one special week for computer campers with diabetes.

The camp medic, Dr. David Rowe, is a pediatrician from the University of Connecticut who works with diabetic children. His interest in diabetic children may spur the special week next summer.

Zabinski appreciates the complications this would involve, but he's confident they can handle just as they first handled the much greater problem of creating a computer camp. ■

French to Make Model III For Radio Shack Europe

Tandy/Radio Shack, Fort Worth, TX, has some good news for European fans of the TRS-80. The firm is completing an agreement with a French electronics firm to manufacture the TRS-80 Model III in France.

Tandy spokeswoman Harriet Rylander said if things go well the French company, Matra SA, will have the microcomputers in production by March.

Tandy is making its French connection to improve the supply of TRS-80s in the Common Market countries. Tandy involvement in the European microcomputer market does not approach the success it has had in the U.S., and Rylander blamed this on lack of supply.

Tandy's U.S. manufacturing facilities are straining to keep up with domestic demand, so TRS-80s have been rare in Europe. The Model III was not available there at all until June, and is still hard to find.

The French connection is the second foreign agreement Tandy has announced in the last few months. They earlier announced an agreement with Tokyo Electric to manufacture and jointly market the Model I in Japan.

Until this summer Tandy had made all its microcomputers in its own plants in the U.S. However, Rylander said the announcements do not so much represent a change of company tactics or policy as the exercising of an option Tandy always considers when supplying a new market. Having a local host simplifies many areas for Tandy.

Other Products

Matra SA is a large French conglomerate active in areas including telecommunications, space technology and mass transportation. According to Rylander, Tandy is interested in developing other connections with Matra, particularly in telecommunications, if the Model III deal works well.

The Matra deal has been approved in principal by the French government. However, arrangements between the two firms have not been finalized, and once the agreement is signed it must be approved in detail by the French government.

Tandy President John V. Roach stated, "This joint venture for computer manufacturing represents an important milestone in our growth in Europe in general and in France in particular. It will significantly strengthen our ability to support the European microcomputer market." ■

School Microcomputer Aid Clipped by Budget Cutbacks

Although a number of federal agencies offer grants that could be used for projects involving microcomputers in the schools, many are receiving funding cutbacks or are being absorbed in the block grant program.

Title IV B and IV C of the Elementary and Secondary Education Act of 1965 authorized the largest sums for microcomputers. The Office of Libraries and Learning Resources, which administers IV B, granted 40 percent of last year's $161,000,000 for projects involving microcomputers. Title IV C, whose funds are granted to improve local educational practice, doled out $66 million in fiscal 1981.

With both funds going to block grants in fiscal 1982, 20 percent of the available money will be given to the states and 80 percent to local school districts, which means the schools themselves will have more control over how the money is spent.

Budget Jitters

Many grant agencies expect cuts in next year's budgets, however. Authorization hearings are still underway in the House of Representatives, leaving the agencies up in the air about how much money will be allocated next year.

No federal agencies are authorized to fund projects solely for microcomputer purchase or use, but many will fund larger programs involving micros. These include:

• The Division of Adult Education: Part of the Office of Education, Adult Ed recently dispersed $100 million in grants to expand adult education skills. Money is given to the states who determine allocations to school districts. The program should remain a categorical grant

See List of Advertisers on page 386

for next year.

• Bureau of Education for the Handicapped, Division of Innovation and Development: The Education of the Handicapped Act authorized money for research into improved education for the handicapped. Some applicants funded in the past already had microcomputers; grants have supported computer-administered testing and research, as well as curriculum development using micros. Max Mueller, chief of the research projects branch, says next year's budget could run anywhere from $8–20 million.

• Office of Bilingual Education, Office of Education: This program is designed to improve teaching English as a second language, using the student's native language. The program funds micros as part of a basic bilingual program for elementary or secondary education. They awarded 525 grants last year and expect a budget of $139,000,000 for fiscal year 1982.

• Fund for the Improvement of Post-Secondary Education: Robert Fullilove, director of the program, says they have granted "substantial funding" for micros at colleges and universities. Grants have been awarded to San Francisco State College to establish a "math anxiety clinic" with a Commodore PET computer, and to the University of California at Santa Barbara, where micros are used in algebra classes. Fullilove estimates level funding for next year, which would be about $13 million.

• Appalachian Regional Commission: Authorized by the Appalachian Redevelopment Act of 1965, this program provides grants to vocational schools and post-secondary technical schools. Director Melvin Rottenberg describes it as a "flexible block grant program" administered by the area states. Their maximum budget for 1982 should be $50–65 million.

• The National Science Foundation: While the foundation has not awarded money for micros in elementary or secondary schools since 1970, it has awarded substantial grants to post-secondary schools for research involving computers. As part of the Local Course Improvement Program (LOCI), they gave $2.49 million to 125 institutions for research; another program awarded $2.7 million in 1980 for the purchase of scientific equipment at 215 colleges. Grants included $18,569 to the University of Hawaii to develop a software laboratory, and $13,975 to SUNY College at Oswego for an on-line microcomputer laboratory for behavioral research. About half of the LOCI awards go to microcomputer projects, says John Maccini, who coordinates the program:

• Teacher Center Program, Bureau of Higher and Continuing Education: Teacher Center grants establish in-service centers for educators; many of these contain microcomputers. The program has a $10 million budget for fiscal year 1982, and will be absorbed by the block grant program the year after. It originally funded the establishment of 60 teacher centers in 1978; half of these have retained funding. Many of the others are now supported locally.

No Funds Here

Programs that are being dissolved or which no longer fund micros include:

• Coastal Plains Regional Commission, a program assisting vocational schools in the southern coastal areas, is ending Sept. 30; it had received applications involving micros but none were approved by the states.

• Emergency School Aid Act of the Equal Educational Opportunity Program cannot allocate its monies for micros, according to director Jesse Jordan. The agency was incorrectly included in a list of available funds collected by Bell & Howell, Chicago.

Even if federal funds are limited in the 1980s, the agency officers doubt this will have a detrimental effect on micro development in schools.

"They are getting computers in the classroom by hook or by crook," says Bob Tinker of the Technical Education Resource Center (TERC). PTAs and districts are being called upon as well as formal state and federal funding sources.

Linda Roberts, a researcher for the Division of Educational Technology, agrees. "School officials are saying, 'Hey, I want a micro—and we're having a cake sale to get it.' There's a real grass roots effort." ■

by Betty Thayer
80 Microcomputing *staff*

Students from 5 to 105 Enjoy Hands On Science

The Talcott Mountain Science Center for Student Involvement (TMSC) in Avon, CT, is an unusual learning institution. Dedicated to teaching science by getting students involved, it educates children from kindergarten through college level in private programs for Center members and by the class under contract to a variety of public and private schools and the University of Hartford in nearby West Hartford, CT. Computer programming is only one of the several hard science subjects the center offers.

While most schools take a classroom approach to science, with a heavy emphasis on lecture and theory and a limited amount of laboratory time, TMSC presents little or no lecture and puts the students into a hands-on situation under the guidance of teachers and scientists. TMSC students have been involved in genuine scientific research, helping staff members develop raw data for papers for international symposiums. The students, who range from age five to 22 and from upper middle class "rich kids" to inner city Hartford blacks and Chicanos, respond enthusiastically. Careful testing shows they also learn more quickly and thoroughly than students in traditional classrooms.

TMSC's Computer Science Department is fully integrated into the general activities of the Center. Housed in a new solar-heated building, the Center offers students a variety of microcomputers which include all the Radio Shack products. Over the summer the heaviest users are members of the Center's Quest program. These are teenagers from grades 5 through 12 with high science aptitudes who work on projects of their own choosing for two week periods. They are divided into beginners, intermediate and advanced or Assembly language classes which work in three separate rooms in small group situations.

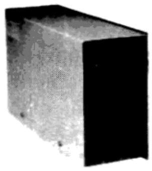
Practical Programs

An unusual aspect of the Center program is that it is self-reinforcing. Many of the students work on programs that they or others at the Center will use to learn about computers or augment other science projects. One student is developing a program which will accept weather data from radio weather reports and draw a weather map. When it is finished the program will be used by the Center in their meteorology courses. Another built his own specialized microprocessor to accept solar energy information directly from the Center's solar measurement equipment and analyze it automatically for use with the Center's continuing program of solar science.

Writing instructional programs is another popular activity. "The kids get a big kick out of teaching other kids something and writing programs that

WE WILL NOT BE UNDERSOLD

DISK DRIVES



FOR TRS-80* Model I

CCI-100	5¼", 40 Track (102K)	**$299**

ADD-ON DRIVES FOR ZENITH Z-89

CCI-189	5¼", 40 Track (102K)	**$394**
Z-87	Dual 5¼" system	**$995**

External card edge and power supply included. 90 day warranty/one year on power supply.

RAW DRIVES

5¼" TEAC or TANDON	**$ CALL**
8" SHUGART 801R	**$ CALL**

DISKETTES — Box of 10

5¼"	**Maxell**	**$40**	**BASF/Verbatim**	**$26.95**
8"	**Maxell**	**$45**	**BASF/Verbatim**	**$36.00**
PLASTIC FILE BOX-Holds 50 5¼" diskettes				**$19.00**
PLASTIC LIBRARY CASE	5¼" **$3.00**		8"	**$ 4.00**
HEAD CLEANING DISKETTE				**$25.00**
FLOPPY SAVER	**$10.95**		RINGS	**$ 6.95**

16K RAM KITS — 2 for $37 — $19

200ns for TRS-80,* Apple II, (specify): **Jumpers $2.50**

SYSTEM SPECIAL

Apple II Plus 48K w/drive and controller. Epson MX-80 printer and interface. SUP-R Mod RF Modulator: List $2965 You Pay **$2299**

COMPUTERS/TERMINALS

ARCHIVES	64K, 2-Drives, 77 Track			**$CALL**
ALTOS	ACS8000 Series			**$ CALL**
ZENITH	48K, all-in-one computer			**$2200**
ZENITH	Z-19			**$ 725**
TELEVIDEO	910 **$559**	920C **$ 729**	950	**$1039**
IBM	3101 Display Terminal			**$1189**
ATARI	400 **$ 359**		800	**$ 795**
MATTEL	INTELLIVISION			**$ 259**
APPLE PERIPHERALS				**$ CALL**

S-100 CALIFORNIA COMPUTER SYSTEMS

MAINFRAME	Model 2200A	**$349**
Z80 CPU	Model 2810	**$259**
MOTHER BOARD	Model 2501	**$106**
16K STATIC RAM, 200ns	Model 2116C	**$309**
32K STATIC RAM, 200ns	Model 2032C	**$619**
64K DYNAMIC RAM	Model 2065C	**$580**
FLOPPY DISC CONTROLLER	Model 2422A	**$345**
EXTENDER BOARD	Model 2520K	**$ 52**
2P + 2S I/O	Model 2718A	**$309**

PRINTERS

NEC SPINWRITER
Letter Quality High Speed Printer

R.O.	**$2395**
R.O. with tractor feed	**$2555**
KSR with tractor feed	**$2795**

NEC SPINWRITER	3500 Series			**$CALL**
EPSON	MX-70 MX-80	MX-80FT	MX100	**$CALL**
PAPER TIGER				
IDS 445	Graphics & 2K buffer			**$ 695**
IDS 460	Graphics & 2k buffer			**$ 799**
IDS 560	Graphics			**$ 999**
ANADEX	DP-8000 **$849**		DP-9500/01	**$1295**
OKIDATA				
Microline 80	Friction & pin feed			**$CALL**
Microline 80	Friction, and pin & tractor feed			**$CALL**
Microline 82	Friction & pin feed			**$CALL**
Microline 83	120 cps, uses up to 15" paper			**$CALL**
CENTRONICS	739, new model with graphics			**$ 739**
C. ITOH				
Starwriter I	25 cps, parallel interface			**$1449**
Starwriter I	25 cps, serial interface			**$1529**
Starwriter II	45 cps, parallel interface			**$1829**
Starwriter II	45 cps, serial interface			**$1949**

PRINTER SPECIAL

SEIKOSHA GP-80M List $399 You Pay **$319**

MONITORS

BELL & HOWELL	9" B & W BHD911			**$175**
LEEDEX	12" B & W	**$129**	12" Green Screen	**$159**
	13" Color	**$329**		
SANYO	9" B & W	**$149**	12" Green Screen	**$238**
	12" B & W	**$219**	13" Color	**$406**
ZENITH	13" Color	**$349**	12" Green Screen	**$129**

TELECOMMUNICATIONS

LIVERMORE STAR MODEM 2-year guarantee	**$125**
UNIVERSAL DATA SYSTEMS UDS-103	**$179**
D-CAT HARD WIRED DIRECT MODEM	**$189**
AUTO-CAT Auto Answer, Direct Connect Modem	**$249**
APPLE-CAT II	**$349**
D.C. HAYES MICRO-MODEM	**$295**
D.C. HAYES SMART-MODEM	**$235**
CCI Telnet Communications Package	**$135**

APPLE ACCESSORIES AND SOFTWARE

VISICALC $155.00 / **DB MASTER**	**$159.00**
CPS MULTIFUNCTION CARD	**$189.00**
Z-80 SOFTCARD	**$259.00**
VIDEX BOARD	**$255.00**
KEYBOARD ENHANCER	**$110.00**
16K CARD	**$159.00**
APPLE JOYSTICK	**$ 49.00**
SUP-R MOD	**$ 25.00**
CCS CARDS	**$ CALL**
GALAXIAN	**$ 22.95**
SPACE ALBUM	**$ 35.00**
ASTEROIDS	**$ 17.95**
FLIGHT SIMULATOR	**$ 29.00**
WIZARD & PRINCESS	**$ 28.00**
SARGON 2	**$ 29.00**
MYSTERY HOUSE $ 24.00 / **HI-RES FOOTBALL**	**$ 35.00**

For fast delivery, send certified checks, money orders or call to arrange direct bank wire transfers. Personal or company checks require two to three weeks to clear. All prices are **mail order only** and are subject to change without notice. **Call for shipping charges.**

DEALER (NATIONAL/INTERNATIONAL) INQUIRIES INVITED — **Send for FREE Catalogue**

The CPU SHOP ✓298

TO ORDER CALL TOLL FREE 1-800-343-6522
TWX: 710-348-1796 Massachusetts Residents call 617/242-3361

5 Dexter Row, Dept. MC10M
Charlestown, Massachusetts 02129
Hours 10AM-6PM (EST) Mon.-Fri. (Sat. till 5)

Technical Information call 617/242-3361
Massachusetts Residents add 5% Sales Tax
Tandy Corporation Trademark/®Digital Research

master charge — VISA

teach," Bill Danielson, TMSC's computer department director, said.

These programs have included mineral and plant classifiers which ask the user a series of identification questions about the specimen and use the answers to determine what it is.

One of the more interesting student projects involved fruit fly propagation. The student wanted to demonstrate the rate of increase of a fruit fly population using the fruit fly propagation formula. Normally it takes two weeks for fruit flies to hatch. The student wrote a program to print a colon to represent each pair of fruit flies. He started with a single pair and had the microcomputer calculate how much time it would take to print out each generation on a 30 cps printer. He discovered that if the computer runs at full speed, the actual fruit flies will catch up with it in their fifth generation and the printer will fall progressively further behind after that.

The nicest thing about this approach to learning is that the students learn programming, use of the computer and computer applications to other activities at the same time they are learning the subjects covered by their programs.

Their learning goes beyond the immediate applications, however.

"One of the real advantages of the computer is it is easy to construe it as a brain," Danielson said. "It gets the kids to think very clearly about who they are and what the nature of thinking is."

"We are no longer a supplement to our students' science education. We are their science education."

The Quest students are not the only Computer Center users. Danielson said the Center has been running a large number of teacher workshops for area schools interested in developing more applications for the microcomputer in their classrooms. The Center also has younger children in programs; the youngest are the "Fledglings," kids going into kindergarten and Grade 1 in the fall. They get one and a half hours with the microcomputers as part of their program of general introduction to science.

The Center relies heavily on its TRS-80s; it has five Model IIIs, one Model I, a Pocket Computer and a Color Computer along with three Apples, one Sinclair and a PDP-11 minicomputer from Digital Equipment Corp. (DEC) that time-shares on several terminals for the most complex applications.

"The important thing is not whether they master Basic or Assembly language but that they have a good experience with the machine at an early age," Danielson said. "With the increasing intrusion of computers into our professional and private lives, we are facing a two-cultures problem. We may well find ourselves divided between those who use computers and those who do not."

"Most of these students won't become

professional programmers; they will go out and be engineers, businessmen, writers, and many other things. If they bring a positive attitude towards computers to their work they will be able to use this tool effectively."

Important Place

In a decade of cutbacks in public education funding, Danielson said TMSC and other institutions like it serve a vital purpose in education. The head of one large Connecticut school system regularly visits public school classrooms on a drop-in basis. In over 100 visits in the last year, he found only two instances where a science lesson was being taught when he visited.

The obvious implication is that only two percent of the students' time is spent on science. In an increasingly technological world, it is ironic that the return-to-basics movement in education is discouraging any change in that proportion.

Under these conditions, Danielson said, "We are no longer a supplement to our student's science education. We *are* their science education." ■

College, Publisher Join, Offer Computer Program

The Hawthorne-Green Inc. Institute of Computer Science has been announced by officials of Nathaniel Hawthorne College. The new program will be offered at the Hawthorne campus, located here in Antrim, a small town in the south central region of New Hampshire.

Initially, instruction will deal with the fundamentals of computing, with a heavy emphasis on microcomputers. The courses will explore hardware, software, and computer science; future plans call for increasingly diverse offerings of advanced and specialized courses.

The classes will be taught by the Hawthorne faculty and by members of the staff of Wayne Green Inc., located in nearby Peterborough, NH. In addition to publishing several microcomputer periodicals, the firm also maintains an extensive working microcomputer laboratory for use by its subsidiary, Instant Software, one of the country's major microcomputer software houses.

In the interest of making the classes available to as many people as possible, the first courses will be held in the evening, two nights a week. Students with no technical background will be accommodated in the introductory sessions as will the needs of more advanced enrollees. Topics to be covered include data processing and electronic trouble shooting and, according to a spokesman for the institute, the classes are well suited for the person who wishes to prepare himself for a career in the booming technical field. Course participants may also elect to pursue a certificate program or go on to further study under a formal degree program.

The first courses are slated to begin this month. ■

Farmers Group Cultivates Videotex News Database

A new timesharing network for farmers supplying up-to-the-minute information on commodities markets has been started by the Professional Farmers of America, Cedar Falls, IA.

The electronic news network, called Instant Update, uses Radio Shack Videotex 16K terminals to distribute a daily morning newsletter, a report on opening prices in the commodities markets, updates on the markets through the day and evening market summaries. It also carries information on market strategies, reports on events in Washington which may affect the commodities market and background and historical information. The network offers analyses of events which may have an impact on the market and their possible effects on commodities prices.

"The network offers analyses of events which may have an impact on the market and their possible effects on commodities prices."

The most recent addition is a database of Gulf Coast grain prices. This is a measure of demand in the market according to Tom McCafferty, marketing manager for the Professional Farmers. He said they were considering adding East and West coast grain prices.

McCafferty said interest in Instant Update has run high among farmers since it was first announced in January. The service actually started April 1. McCafferty declined to give the number of subscribers except to say it is "over 500." He said while subscribership is heaviest in the midwest, Instant Update is a nationwide service which even has a subscriber in Hawaii.

It costs a flat $95 a month for unlimited access, including rental of the Videotex terminal. It is an interactive service, with the user selecting a database from a menu and storing information in Videotex to read off the television screen.

McCafferty said the newsletter and commodity prices are the most used databases, a weather database the least.

Only Choice

The Professional Farmers chose Videotex because it was the only terminal around. They have been very happy with the equipment which so far has been virtually trouble-free.

McCafferty said relations with Tandy/Radio Shack, Fort Worth, TX, have been good although "they can always be better." He also said they have been waiting for some software from Tandy to allow TRS-80 microcomputers to interface with the system.

Charles Phillips, Senior Vice President for Special Markets for Tandy, mentioned they are actively promoting the idea of private information networks using Videotex terminals. In the long run, he said, Tandy expects Videotex to become a household appliance. Phillips told us they hope enough private networks will take up Videotex to create a "critical mass" of homes using it so that as general-interest data services develop they will be designed to be compatible with Videotex. "A lot of people are looking at this form of communication," he noted.

McCafferty said Professional Farmers is a private business offering farmers marketing information and seminars on marketing. Founded in 1973, it has more than 30,000 client-members. ■

Education Bill Reintroduced

US Rep. Tom Downey (D-NY) has reintroduced his bill to establish one or more national centers of study microcomputer use in education. The bill is being co-sponsored by two members of the House Subcommittee on Elementary and Secondary Education, where it stalled last year. According to a spokesman, Rep. Downey is optimistic that subcommittee hearings will be held on the bill this year.

The bill seeks $4 million paid over three years to support the establishment of one or more centers to study how microcomputers are used in schools and to make this information available to interested parties in the hope of encouraging microcomputer use in education. However, David C. Smith, legislative assistant to Rep. Downey, warned that it will be tough to squeeze extra money out of Congress for anything this year.

Smith said one big argument for the bill is that the United States is falling behind other countries in this area; even the British, with their much-publicized economic woes, are spending $25 million to purchase microcomputers and put them in every school in the country. The French have a similar plan already approved and in operation.

"All we want to do is study the uses, we aren't talking about actually buying any machines for schools," Smith said.

Rep. George Miller (D–CA) and Rep. Mario Biaggi (D-NY) are co-sponsoring the bill this year. Smith said the hope to conduct subcommittee hearings sometime after the August recess.

If the subcommittee passes it, the bill will then be considered by the Committee on Education and Labor. If that committee acts favorably on it, it will go to the full House for consideration.

Anybody interested in expressing an opinion on the bill, labeled HR-2112, to be placed on the record, may write to Rep. Carl Perkins (D-KY), Chairman, Subcommittee on Elementary and Secondary Education, Room B346C, Rayburn House Office Building, Washington, DC 20515. ■


ATARI 400 COMPUTER TO BE AWARDED TO ESCAPEE!
ASYLUM
Asylum is the most sophisticated, sinister, challenging 3-D graphics adventure ever written! You are placed in an asylum for DEATHMAZE survivors. Your only goal: *ESCAPE!*
Asylum has over 1200 locations! Hallways recede into the screen, *as though you are actually there!* Doors open and close! Beds, desks, and even guards and other inmates are drawn on the screen! And graphics are generated instantaneously! Simply press an arrow key to move or turn, and you are moved. Absolutely *no* waiting!
English language commands may be issued at any time from a vocabulary of almost 300 words. Commands may include complete statements. *"Get everything on the desk except the matches"* is perfectly legal!
Will anyone escape? Who knows? Intelligence, cunning and patience will be necessary. Send us your correct solution, postmarked no later than November 15th. On November 30th we will draw from the entrants for an ATARI 400 16K color computer!
NO HINTS WILL BE GIVEN OVER THE PHONE!
ASYLUM 16K cassette $ 19.95
ASYLUM 32K disk $ 24.95
ASYLUM Hint Sheet $ 1.00
TRS-80 Level II, Model I and Model III only
128
Med Systems Software
P.O. Box 2674-A
Chapel Hill, NC 27514
(919) 933-1990

It's Not Your Printer, It's Your Software.

Do you own a Centronics 737? 739? Lineprinter IV? Tired of the limitations of SCRIPSIT? **Qwerty 3.0** will let you accomplish word-processing feats you never dreamed possible. Like Greek letters. Simultaneous subscripts and superscripts. Even footnotes and two or three column folio formats. See the list of features in the box. No other piece of software of this type can match **Qwerty 3.0**. We know, because we bought the others, and were disappointed.

Qwerty 3.0 is more than "fully tested." It has seen thousands of hours of use in a university environment. A masters thesis and a statistical doctoral dissertation were produced and accepted right off a Centronics 737 using this package.

Qwerty 3.0 is the finest SCRIPSIT addition available for the Centronics 737, 739, and Lineprinter IV. You must own one of these printers, a copy of SCRIPSIT, and a disk drive to use it. It is the best. Period. WE GUARANTEE YOU WILL AGREE! If for some reason, you find that this program doesn't meet your needs, return the entire package within 14 days for a prompt and cheerful refund.

For cautious buyers, we offer the manual (about 50 pages) for $10.00. When you decide to buy **Qwerty 3.0**, we will credit the full manual price.

Features of Qwerty 3.0

1. Automatically prints in proportional print, with a suitable format. Transitions between the three print styles are easy, including all expanded print modes.
2. **Qwerty 3.0** adds 75 new symbols, including upper and lower case Greek letters, mathematical symbols such as integrals and summations, arrows, brackets, and probability symbols.
3. Any character can be used as a subscript or superscript, even simultaneously. Carats, bars, and tildes can be placed over any character, with precise position control.
4. Underlining, with or without underlining of spaces, including long ratios and mathematical expressions.
5. **Footnotes** can be placed on any page so that they remain on the desired page, even if text is inserted later.
6. **Table** commands enable positioning of the print head anywhere on a line. Invaluable in printing neat mathematical layouts, tables, columnar material, etc., in proportional print.
7. **Pretty** commands allow printing of repetitions of a chosen character. When combined with **Table**, decorative borders can be produced with ease.
8. **Folio** format produces output in two or three columns per page, in either proportional or 16.7 cpi mode. Ideal for newsletters.
9. Supplies a third output mode, in which only **Scripsit** commands are obeyed. Allows printing of special **Qwerty** commands for future reference.
10. **Page End** indicates where pages will end, and the page number, without printing the text. One can prepare an almost error-free document without ever using paper.
11. Correction of **Scripsit**'s errors and inconveniences, extensive documentation, and much more!

Qwerty 3.0 Disk, Manual, Reference Card, and Printer Table Rule $49.95

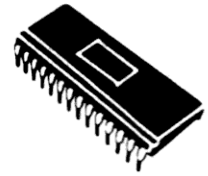
Qwerty 3.0 Manual Alone $10.00

Med Systems
(919) 933-1990

128

More from Med Systems

To Order Call 1-800-334-5470



Microworld
by Arti Haroutunian

Microworld is an adventure like no other. You are transformed into an electroid, and must travel through the circuits of your TRS-80! You will be required to solve many incredible problems, and in the process, you will gain a knowledge of how a TRS-80 operates.

Microworld is one of the most advanced pieces of machine language programming Med Systems has published. A special encoding scheme has allowed a 21K adventure to fit in 16K. Microworld is verbose. Messages are frequent and fact-filled.

Microworld is an excellent educational simulation. It is supplied with a 12 page booklet containing a glossary and explanations of the electronics inside the TRS-80.

Microworld is immense! It is twice as large as most adventures, and at least twice as challenging!

Tape $19.95 Disk $24.95

Educational Software

The Playful Professor
A mathematics tutoring program that combines graphic step-by-step tutorials with an interesting graphics game. Tutors all operations, with integers or fractions. One or two players, three difficulty levels.

The Human Adventure
A biological adventure in the human body. The player travels through the cardiovascular system, learning locations and functions of bodily organs while pursuing cancer cells. Grades 4 and up.

Money Master
A graphics game which sharpens skills in counting money and making change. The correct amount, in a reasonable number of coins, buys objects and pays tolls. Some reading required.

Educational Software
Tapes $14.95 ea. Disks $19.95 ea.

Educational Package Disk
All three Med Systems educational programs on one disk **$34.95**

Med Systems
P.O. Box 2674-A
Chapel Hill, N.C. 27514
(919) 933-1990

128

Deathmaze 5000 Labyrinth

Deathmaze 5000 and Labyrinth are 3-D adventures from the authors who produced Asylum. One and two word commands are combined with incredible 3-D graphics to provide adventures beyond anything you've ever seen.

The goal in Deathmaze is simple. Escape from the most maniacal, devilish, dangerous building ever constructed. Alive.

And Labyrinth? Kill the minotaur. However, you must find weapons, treasures, and knowlege to do so. There's even a graphic vending machine!

The question isn't "Will I survive?". The question is "Will I be satisfied with non-graphics adventures again?".

Tapes $14.95 ea. Disks $19.95 ea.

GRBASIC
by Simon Smith

GRABASIC extends Level II or DISK BASIC to include an easy to use graphics command set. A single BASIC command allows the user to draw a line between any two pixels on the screen in hundredths of a second! Coordinates can be chained to allow complex figures to be drawn by a single BASIC program line in less than a second!

GRBASIC allows the definition of shapes. Once defined, a shape can be *rotated*, *scaled* up or down in size, drawn anywhere on the screen in less than a second, and can even be drawn totally or partially "off" the screen in extended space! And all with short, simple BASIC commands! Even multiple shapes are no problem!

A new program, GRBASIC FUNCTION PLOTTER, allows the plotting of almost any function, including polar coordinate based figures, parametic equations, and almost any wave form. Features include function definition and automatic screen scaling, and printing capacity on Line Printer IV. REQUIRES GRBASIC!

GRBASIC Tape	**$19.95**
GRBASIC NEWDOS Disk	**$24.95**
GRBASIC TRS-DOS Disk	**$24.95**
GRBASIC Function Plotter Tape	**$14.95**
GRBASIC Function Plotter Disk	**$19.95**

Knossos
by Simon Smith

"You have just been thrown into a dark, stinking pit . . ."

You are isolated and alone in the maze at Knossos, Crete. Somewhere, a minotaur is tracking your scent. Can you find the only door without becoming a minotaur's meal?

Knossos is a 3-D graphic simulation. Mazes are represented by a perspective view, *as though you are actually there*. These graphics are not the simple, square graphics you have seen before. An entirely new representation has been implemented giving a true cave-like quality. And like all Med Systems 3-D graphics, lightening fast screen generation is standard.

Other features include chalk with which to mark the floor for reference points, randomly generated mazes, distance counters for exit, and monster graphics. A typical game might last 15-20 minutes. This is the first *truly* 3-D arcade game ever offered.

Tape $19.95 Disk $24.95

Satisfaction Guaranteed
If for any reason you are not satisfied with our products, return your order within 14 days for a prompt and cheerful refund.

Ordering Information
Orders are processed within five working days. Orders are sent first class within the U.S., Canada, and U.S. territories. Overseas orders please include $3.00 for air post.

Order using coupon below. Please specify tape or disk versions, and Model I or Model III computer.

Model I or Model III TRS-80 Level II 16K unless otherwise noted

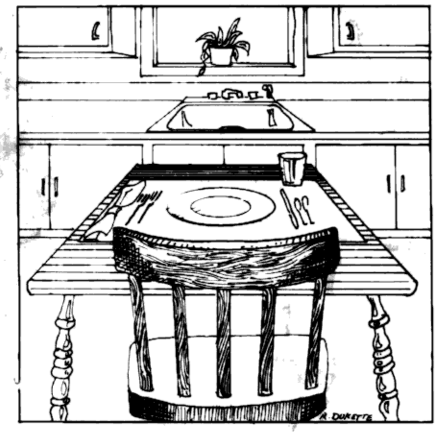
Item (specify tape or disk)	Price
Shipping and Handling	$1.00
TOTAL	

Name ____
Street ____
City ____ State ____ Zip ____
Computer:
TRS-80 Model I ☐ Model III ☐
Mastercard ☐ VISA ☐ Check ☐
MC or VISA # ____
Expiration Date ____

See List of Advertisers on page 386

News From KITCHEN TABLE SOFTWARE, INC.

by David Busch



What do you get when you take features from both Basic and Cobol? A new language called Basbol. This innovative high-level language, a combination of ANSI '74 Cobol and Dartmouth Basic, has been introduced for the TLS-8E by Kitchen Table, Inc.

Regular *80 Microcomputing* readers will recall that KTI is the United States' largest fictitious supplier of space-age computer products, including DROSS-DOS 1.1 and the TLS-8E. The latter is a new personal computer that is 100 percent downward compatible with the TRS-80 Model I and III.

Basbol was one of six new products unveiled recently by Kitchen Table at the Portage County (Ohio) Home Appliance Fair. Also making their U.S. debut were a new 5 3/8-inch disk drive, a double-density board and a series of aids designed to treat radiation burns stemming from an initial run of poorly-designed color monitors for the TLS-8E.

Biggest hit of the appliance show, at which KTI was the only computer representative, was the Basbol exhibition. The new language is a compiler/interpreter that is used in a unique way. Very fast-running object code is generated for rapid testing (although debugging is impossible) and then converted back for run-time operation under the Basbol interpreter.

KTI spokespersons pointed out that this mode eliminates many errors, because each line of code is interpreted every time a program is run as a sort of double check. At the same time, the compiler (interpreter) prevents unwanted tampering with the code by the programmer at the testing stage.

A Controversy

One note of controversy surfacing at the display was that Basbol is supplied on a protected disk. The original disk can be backed up only 10 times. It then becomes useless for copying purposes. This will probably not cause much distress for the legitimate user because each backup disk also can be duplicated 10 times with no problem.

We suspect this flimsy attempt at protecting Basbol was a subterfuge on KTI's part to make the new software more attractive to program pirates. Much of the firm's influx of working capital stemmed from the widespread copying of DROSS-DOS 1.1. Though KTI sold less than 300 copies of this DOS over the counter, it has reaped more than $3 million in profits marketing mandatory Zaps to 111,306 users nationwide.

"The original disk can be backed up only 10 times."

Our suspicions were confirmed on examining the documentation provided for Basbol. It is printed in blue ink on a dark green paper—all but unreadable in original form, but eminently legible when photocopied.

Other products also attracted a lot of attention. KTI's booth was mobbed during exhibit hours. The Da-Glo orange TLS-8E computers really stood out among the drab Harvest Gold refrigerators and home freezers in the surrounding booths.

New Drives

The product that interested me most was the 5 3/8-inch disk drive now available for the TLS-8E. These new drives replace the units originally supplied with this computer. Scuttlebutt has it that KTI was experiencing delivery problems with the original drives—customers would take one look at the units and refuse to accept delivery.

Because KTI has been having trouble getting its Sri Lanka factories to adhere to strict specifications, the new drives will accept either the common 5¼-inch mini-floppy disks or the occasional 5 3/8-inch ones that slip through inspection. The KTI-100 units are 96-tracks-per-inch drives that can be run either in single or double density. Double-density reliability, using the new KTI Doubloon PC board, is said to be 100 percent that of the TLS-8E in single-density mode. Company spokesmen have promised a remedy for this unfortunate situation in the near future.

The KTI-100 drives may be configured as fixed disk drives merely by inserting a disk and applying furnace ductwork tape over the door to seal the unit shut. Quite a clever touch.

Since the introduction of the TLS-8E, TRS-80 owners have written to comment. One frequent question is, "So what?" A more probing query runs along the lines of "How can a TLS-8E benefit me, a proud Model III owner?"

So What?

My standard answer has been as follows:

- KTI's marketing of the TLS-8E will put a great deal of pressure on Tandy to shape up. With three or four hundred thousand Radio Shack computers already in the field, Tandy has great expectations of selling peripherals, programming, fuses, etc. to this large customer base. Given sufficient provocation, it's conceivable that a quarter of a million TRS-80 owners might decide to switch to TLS-8Es overnight. Then where would Tandy be? Back selling leather belts, that's where!
- Because most software available for the TRS-80 can be used on the TLS-8E, this provides a larger customer base for programmers, and, correspondingly, more software available for the rest of us. Instead of 400,000 Tandy customers, a pro-

grammer has a more attractive market of 400,367 users when TLS-8E owners are lumped in.

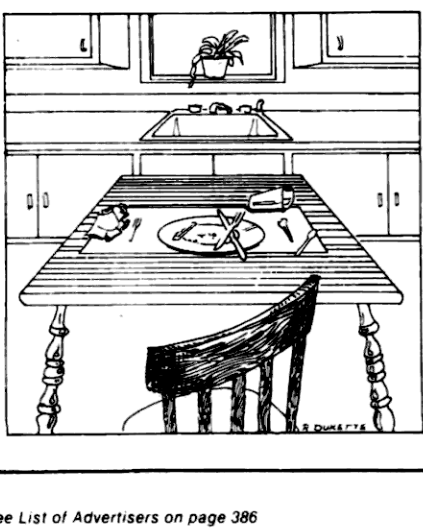
● The TLS-8E provides an attractive "second" computer for present TRS-80 owners. Recent upheavals in the international money market have given the U.S. dollar a stronger position when compared to Sri Lankan currency. As a result, KTI has been able to drop the price of the TLS-8E from $1984 for a two-disk 48K system down to $179.95.

Obviously, at that bargain price, many TRS-80 Model I or Model III owners are going to want a TLS-8E to carry around to users' groups, on camping trips, etc., where its rugged hard-wiring will better withstand mistreatment.

". . . the remains can be rewired to make a pretty good garage door opener."

I've heard of many hobbyists who have purchased these machines just as "parts" computers. Need a 1771 chip? Rip it out of the TLS-8E. Interested in finding out if the Z-79A chip would make a good diode? Use one of the two piggybacked into every KTI computer. After you've experimented to your heart's content, the remains can be rewired to make a pretty good garage door opener.

I gleaned all these tidbits from brief interviews with the reclusive founder of Kitchen Table Inc., who prefers to remain anonymous. We agreed to abide by Scott Nolan Hollerith's wishes and pass along these interesting comments to you without any attribution. ■



THE EXCLUSIVE ORACLE

by Dennis Kitsz

"I have yet to see you name programs that are no good. You are culpable in not disseminating this information. Must we continue to get ripped off?"

Q. I recently wrote a Basic program with machine language interfaced within it. I found it was faster to save the Basic part of the program together with the machine language routines by using T-Bug, than to load them separately or to POKE them in. The program works perfectly when saved this way except for one thing: In order to list the program, it is necessary to type Clear and hit enter before typing LIST. If you don't the computer will List the first line and stop. If you run the program and then List it, it works fine.

My questions: Why is it necessary to type Clear, can you get around it, and why is it necessary to type ?MEM after returning from some machine-language programs? I know that this eliminates errors that occur if you don't, but what causes these errors in the first place? Also, is there any way to get a Basic program to run from a machine language introduction? I have a program written, but the introduction is in machine language. Is it possible for the machine language routine to start the Basic program running when the introduction is finished?

Nathan W. Harrington
Nelson, Nebraska

A. Welcome to Software Interfacing 101: Getting in and out of Basic with no help from Fort Worth. Your first question brings up an interesting topic. The Clear subroutine is used quite often by Basic—after executing Clear, of course, but also after New, CLOAD, and Run. Its purpose is not only to clear out all variables (actually this is a consequence of the routine, not its main objective), but also to reset properly all the line pointers in the Basic listing. What are line pointers? Basic picks its way through a program listing in two ways—by knowing a line's number (stored in binary) and also by knowing ahead of time the memory address of the next response line in the program. This memory address is its "pointer".

The Clear routine automatically sweeps through the entire Basic program, adjusting line pointers according to where it finds the ends of lines. Depending on the condition of the Basic program when you first saved it to tape using T-Bug, your subsequent load attempts may actually be putting it in an area different from the one you expect. Hence, there is an incorrect or truncated list. In your case, the line pointer that should be locating the second line of the program is instead pointing back to the first line. Since Run also uses the Clear routine first, that explains why the program is automatically corrected by running it. The address of the Clear subroutine, if you wish to use it before returning to Basic, is 1E7A hex.

The reason you have to type ?MEM (or something similar) when returning to Basic from some machine language programs is because many programmers took Radio Shack at their word. In the original Editor/Assembler manual which so many programmers used, return to Basic was listed as address 1A19 hex. However, that address works only under certain conditions; registers have to be loaded and the stack pointer set up for re-entry without an ?OM Error message being printed upon entering the first Basic command. A better Basic re-entry point is 06CC hex, which avoids both the ?OM Error message and rebooting a disk system.

Finally, there is indeed a way to get a machine language routine to start your Basic program automatically. Here's the order: register HL is loaded with 41E7, just before the beginning of the keyboard input area. Memory location 41E8 is filled with Run token 8E, and location 41E9 is filled with line terminator 00. Next jump to the master command execution routine, 1D5A. This is how it looks in assembly language:

```
LD     HL,41E8H     ; Get line buffer location
LD     (HL),8EH     ; Place RUN command token
INC    HL           ; Move to next buffer place
LD     (HL),00      ; Close out the line buffer
DEC    HL           ; Move back in the buffer...
DEC    HL           ; ...to before the beginning
JP     1D5AH        ; And bow to master executor.
```

Q. I've got a Model I Level II 16K machine with a problem. The computer runs fine until I run this program:

```
10  READ A,B,C
20  PRINT A,B,C
30  GOTO 10
40  DATA 1,2,3,4,5,6,7,8,9
```

When I run this program I get:

```
        1   2   3
        1   2   3
        1   2   3
etc.
```

I've eliminated the Z80 and eight 16K chips by replacing them. I've had this computer since 1978 and I've installed a lowercase modification in it. I own T-Bug and TLDLS but I haven't figured out how to use them yet. How can I figure out which chip is bad?

SSgt Terry L. Kuns
APO New York

A. The bad chip is your Level II ROM. However, don't replace it as the electronics aren't bad. There's a bug in the first version of Basic. It's too bad you paid for a new Z80 and 16K memory to find that out. Here's the solution. Whenever you are using any program that does tape I/O or performs Read/Data combinations, be sure this line appears somewhere at the start: POKE 16553,255.

Q. I appreciated your January column about the programs you have found to be useful. But, I have yet to see you come out and specifically name programs that are no good. Surely in finding the good stuff, you have found a lot of junk. I feel you are culpable in not disseminating this information, especially when you have written about the stuff that is good. If the authors, such as yourself, who have an audience don't tell us, then must we continue to take a chance and continue to get ripped off?

Larry Morgan
College Place, Washington

A. This column might seem an unusual place to answer Larry's letter, but I've received a lot of mail from TRS-80 users who feel they have been burned. Besides, bad software qualifies as a TRS-80 problem (see the answer to SSgt Kuns's letter above). Okay, there are enormous numbers of programs which I believe qualify as junk—some are given away, some are sold for outrageous prices, and some appear right in the pages of this magazine. Were I to name them, however, I would be overwhelmed with mail from people for whom such programs serve a valuable purpose. On the other hand, even the best of programs aren't good enough for those born of IBM.

Okay, cop-out, you say. But really, what *is* a good program? Perhaps the thousands of meticulously crafted games of video violence which are published and sold? Not for my taste; it seems Klingons are only what you want to call them. How about compilers and other software with the name Fortran? As far as I'm concerned, keep 'em. I use Fortran as much as I use my years of high school Latin. And if I ever see another universal calendar, hex-decimal converter, Basic word processor, telephone dialer, checkbook minder, or alien invader game—I'll scream.

But does that mean I should pan one of these programs and damage both the author's credibility and pocketbook? You don't know me or any reviewer by more than what we write, and that's hardly a keyhole into our likes and dislikes, our capabilities and shortcomings. The best helping hand I can give is to tell you what I feel *is* good. Besides, if I believe it's awful, I'm probably not going to use it long enough to tell you why.

(For a complete list of software that I hate, please send a self-addressed, stamped envelope, together with $5,000, to. . . .)

Q. I hate to bother you again. I have had a video bug in my '80 for as long as I've had the computer (D board, bought used a year ago). When I have much of the screen whited out, the border around the portion of the screen that is not whited out warps, as in the sketches in Fig. 1.

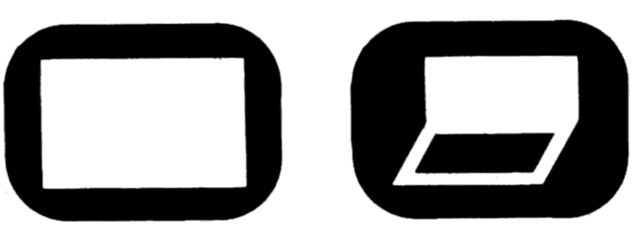


Fig. 1 Warped Screen

When the entire screen is whited out, it has nice square edges, but when a black rectangle appears inside, the white outside the border warps. It interferes with a Breakthrough game I just purchased.

Men Del Cooper
New York City

A. The problem is inherent in the video monitor, which is little more than a stripped television with repackaging (there are dial holes under that TRS-80 insignia) and a little electronic tweaking. Different generations of monitors were of different quality. Open up the monitor (with it unplugged, of course!) and look inside for the vertical circuit board or boards. If you have a single vertically mounted board, find resistor R????. Piggy back on it a 3,300-ohm, five-percent resistor (orange-orange-red-gold). You're all set. If you've got two vertical boards, or you want an even better fix, Archbold Electronics (10708 Segovia Way, Rancho Cordova, CA 95670) has an excellent video perk-up board for about $11, with full instructions.

By the way, Men Del's balky power supply (August) turned out to be caused by a corroded screw clamping the power transistor against the circuit board, causing the voltage to go off intermittently. Emery paper and a hefty screwdriver cured the problem completely.

Q. Just what is the purpose of the terminating resistors in TRS-80 disk drive 21-1160? I have two Radio Shack drives and a four-place Radio Shack cable. I'm planning to get an additional non-Radio Shack 40-track drive. All the ads claim that their drives are compatible with the TRS-80, but when asked where the drive is placed relative to the Radio Shack drives, the advertisers avoid the question. Can you use the same Radio Shack cable for the non-Shack drives? If I wanted to send my Radio Shack drives in for repair, could I use the non-Shack drive alone? If I use all three together, where do I put the non-Shack drive? Do I have to advise the seller of the non-Shack drive that I intend to use it with Radio Shack devices?

Lawrence E. Pyle
No Return Address

A. The hardware aspects of combining 35 and 40-track drives are not complicated, but let me first explain the purpose of those mysterious terminating resistors. Digital electronic circuits come in various families to suit electronic and environmental needs. When long signal lines are involved, it is often wise to use circuits which are activated at the far end of the signal path. It is that kind of "open collector" circuit (see August Applications for a fuller explanation) which is used to activate the disk drives. In order for this circuit to respond, however, the open path must be closed. The termination resistors perform that function. One set is enough as too many can be damaging.

If you have a Radio Shack disk drive cable, the drive definitions (0, 1, 2 and 3) are built into it. Other drives will work fine with that cable, but be sure to explain carefully to your vendor that you are purchasing a third drive (called drive 2) and wish it shipped without terminating resistors and with the select jumper properly wired. If you've settled on a good retailer, you will have no problem with this.

If you have to switch drive positions, you might want to follow another course. Your most flexible drive should be in position 0, because this drive should be able to read all 40 tracks on the disk. It gets more work to do than the rest in

checking directories, reading, saving, formatting, and such.

It gets tricky now. If you are using TRSDOS, you have essentially a 35-track DOS which will ignore the extra tracks on your new drive. NEWDOS was one of the first to handle 40-track drives, and virtually all new disk operating systems can do that and more (LDOS claims it can mix'n'match). First, decide if you wish to access those additional five tracks; if so, check *80 Microcomputing*'s recent DOS issue and consider an operating system other than TRSDOS. When you have obtained it, check to make sure your Radio Shack drives are 35-track drives. Many were specified as 35-track units to maintain compatibility, but are actually 40-track devices.

If you have only 35-track Radio Shack drives, then I would suggest your new 40-track drive be wired for position 0, with the terminating resistors installed, and that you move the 35-track Shack drives to positions 1 and 2. Remove the terminating resistor package from the Radio Shack drive by unscrewing the cover and finding the socketed, integrated circuit-like package (usually colored blue and sometimes marked 1E) and pulling it out.

You may also have to rewire the drive-select jumper (position 1F), though this is unlikely. If the DIP shunt is intact, leave it that way and depend on the Radio Shack cable to make the drive selection. If some shunt bars are broken, then you will have to reconnect the ones you need, and break the others. Pin 1 connected to pin 14 selects drive 0; pins 2 and 13 select drive 1; pins 3 and 12 select drive 2; and pins 4 and 11 select drive 3. Your TRSDOS manual has all the circuit details.

Updates

Many readers have asked how Z80 mode zero and mode two interrupts could be used with the TRS-80. I thought about the problem and came up with a typically mundane solution involving piggybacking an integrated circuit and running a wire or two. Everett B. Ogden of Delmar, New York, had a better idea:

> There are at least two easy ways to use these interrupt modes, one of which requires no internal modifications.
>
> The internal mode is cleaner: if you never intend to use the Test input, cut it loose from Z53 pin 4 and connect the INTAK output to that pin.
>
> Those who don't want to attack their keyboards can tie INTAK to Test on the edge connector. At first glance this would seem to generate a bus request, which has priority over interrupts. Studying the Z80 timing diagrams will show that this doesn't happen. The Z80 samples the bus request input only at the start of the last T-state of any M cycle (T4 for an M1 cycle). INTAK, the combination of M1 of IORQ, ends at the start of T3, so the Z80 does not see it on the BUSRQ line. The messy part is that the Test input also tri-states the address lines, In, Out, RD, WR, RAS, CAS and MUX. The INTAK pulse is short enough that refreshing isn't a problem, and if the In, Out, RD and WR lines are pulled high, it doesn't matter what happens to the address lines. Resistors could be used, but tri-state buffers are better.

Special thanks to Mr. Ogden for a remarkable piece of insight.

Readers please note that Radio Shack has discontinued the National Semiconductor data books. I highly recommend these references (especially the TTL and memory books), which are available from National Semiconductor (2900 Semiconductor Drive, Santa Clara, CA 95051), Digi-Key Corporation (Hiway 32 South, Thief River Falls, MN 56701), and other distributors. ■

Desperate? Send your questions on Model I, Level II, TRS-80s to: Dennis Bathory Kitsz, Roxbury, Vermont 05669

✓See List of Advertisers on page 386

NEW PRODUCTS

Edited by Janet Fiderio

Little Big Lisp

Little Big Lisp, for the Model I, is available from the Department of Computer and Information Sciences at the University of Oregon.

Lisp includes the Lisp interpreter, a Lisp compiler, a trace package for monitoring the execution of programs, a Lisp structure editor, the RLISP programming language which provides a high level language interface to Lisp which resembles Algol and Pascal, and some sample programs in Lisp and RLISP.

The system requires a minimum of 32K and is available on a 5 1/4-inch disk complete with documentation for $30. Complete source listings are available on magnetic tape for $100. For more information contact Jed Marti, Department of Computer and Information Science, University of Oregon, Eugene, OR 97403.
Reader Service ✓167

Color RAM/ EPROM Cartridge

The CMEMORY plug-in cartridge for the Color Computer can supply the user with up to 8K of continuous memory which can be divided into any combination of 2K blocks of RAM Memory and/or 2176 EPROMS. This product allows you to save frequently used utilities or games in easy-to-use cartridges.

CMEMORY uses the unused address space $C000 to $E000, normally used for plug-in game cartridges. By adding a jumper, the computer can be set to automatically execute a program in EPROM whenever the Reset button is hit.

The CMEMORY cartridge sells without any memory for $24.95. 2K RAM chips are available for $19.95 each, 2K 2716 EPROMs for $14, or you can use your own memory. Available from Micro-Labs, Inc., 902 Pinecrest, Richardson, TX 75080, (214) 235-0915.
Reader Service ✓338

S.A.T. Preparation Series

Krell Software has expanded its College Board S.A.T. Preparation Series from five to 25 programs. It now includes sections on word relationships, vocabulary, sentence completion, reading comprehension and mathematics.

The programs are based on past examinations and present material of the same level of difficulty and in the same form as used in the S.A.T.s.

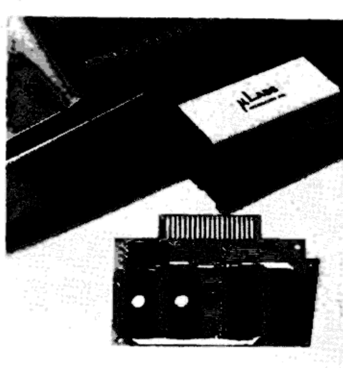
These programs are available for the TRS-80 for $299.95, from Krell Software, 21 Millbrook Dr., Stony Brook, NY 11790, (516) 751-5139.
Reader Service ✓331

Color Computer Game

Packet Man, a popular arcade game, is now available for the TRS-80 Color Computer.

Written in machine code, this game provides the graphics of the Color Computer with the speed of machine code.

The cost of the program is $24.95. A Level II 16K machine is required. Inquiries should be sent to American Business Computers, 118 South Mill St., Pryor, OK 74361, (918) 825-4844.
Reader Service ✓333



The CMEMORY Cartridge

Programs for Beginners

Programs for Beginners on the TRS-80 is written for the computer novice. Written by Fred Blechman and published by the Hayden Book Company, the book provides instruction through 21 programs that run on a Model I and Model III TRS-80.

Programs include topics concerning business, bookkeeping, calculating loan interest, mortgage payments, investment evaluation, and others. The four appendices include a video display worksheet, cassette-loading time charts for Level I and II, and a complete description, schematic and parts list for an audio/visual control box.

This 150 page book costs $8.95 from Hayden Publishing Co., Inc., 50 Essex St., Rochelle Park, NJ 07662, (201) 843-0550.
Reader Service ✓162

D-CAT from Novation

The D-CAT from Novation is a directly coupled modem that is FCC approved for handset jack connection with any modular phone. It operates with either single or multi-line phones without the need for adapters.

D-CAT transmits data over telephone networks allowing one computer or terminal to talk to another. The direct connect feature eliminates distortion and lost-data problems.

It is available for $199 from Novation, 18664 Oxnard St., Tarzana, CA, 91356, (213) 996-5060.
Reader Service ✓349

Lazy Writer

Lazy Writer, a word processing system, is available in a new Model III version and an enhanced Model I version 1.8.

The program is capable of reverse (hanging) indents, superscripts and sub-

scripts, offsetting text to the right, precise printing from cursor position, and printing chained files.

Lazy Writer also provides a communications program that works with a modem. The Model III version has an enhanced communications package, allowing stored characters to be sent with a single key stroke for log-ons, provides a "local" echo, and will receive a full eight bytes.

Lazy Writer is priced at $125 for the Model I and $175 for the Model III. It is distributed by Soft Sector Marketing, 6250 Middlebelt Rd., Garden City, MI 48135, (313) 425-4025.
Reader Service ✓336

QSO Log for Ham Radio Operators

QSO Log, from Manhattan Software is directed at amateur radio operators.

When a call sign is heard on the air the operator using QSO Log enters the sign into the computer. QSO Log will list call, name, QTH, date, time, band, and notes on conversation and on the contact's equipment.

The program dumps to tape, loads to tape, and allows on-screen review of all QSO records. It also allows updating, editing and deletion of entries and printout.

The cassette version (16K or 32K) will run on the Model I or III and is priced at $14.95. For additional information contact Manhattan Software, P.O. Box 1063, Woodland Hills, CA 91365, (213) 704-8495.
Reader Service ✓339

Agricultural Software

Two new programs available from Agricultural Software Consultants, require a Model I or Model III with one disk drive, 32K memory and an 80-column printer.

The first program, FEEDDATA, is designed for feed mills and nutritionists. It stores and retrieves feed data on disk and calculates the composition and nutritional value of any combination of these feeds. FEEDDATA is priced at $60.

MIXIT-2 is designed for feed mills, feed lots, dairymen, and poultrymen to determine the "least cost" for any feed mix and more. MIXIT-2 is priced at $200.

For more information on these two programs, contact Agriculture Software Consultants Inc., 1706 Santa Fe, Kingsville, TX 78363, (512) 595-1937.
Reader Service ✓348

Magnetic Media Preservers

Shield your Magnetic Media

Durable metal cases which protect magnetic media from magnetic fields that cause degradation and erasure are now available.

These metal cases are magnetic media preservers designed for carrying or storing. Models include cases for the 5 1/4 inch and 8 inch flexible disks. A cassette tape preserver is also available.

For more information contact the Magnetic Shield Division of Perfection Mica Company, 740 N. Thomas Dr., Benville, IL 60106, (910) 766-7800.
Reader Service ✓341

The Boss

The Boss is an accounting software package which has integrated all accounting functions into one program. This software offers capabilities such as comprehensive financial ratio analysis, loan payment calculations, amortization schedules, depreciation schedules and statement of changes in financial position.

It has combined accounts payable, accounts receivable and general ledger so they can be accessed at any time during the program run. The Boss runs under CP/M.

For more information on this accounting management tool, contact Lifeboat Associates, 1651 Third Ave., NY, NY 10028, (212) 860-0300. The Boss is priced at $2,495.
Reader Service ✓350

Exam Preparation Programs

A series of Competency Exam Preparation Programs are available from Krell Software.

These programs consist of simulated exam modules, a diagnostic package, and a complete set of instructional programs. They are designed to teach concepts and operations, provide drill and practice, and to assess achievement levels. The series provides a curriculum encompassing mathematical, reading and writing instruction.

The CEPS are available in two software formats: the National Proficiency Series, and the NY State Regents Competency Test Preparation Series. Both are priced at $1,299. For more information contact Krell Software, 21 Millbrook Dr., Stony Brook, NY 11790, (516) 751-5139.
Reader Service ✓332

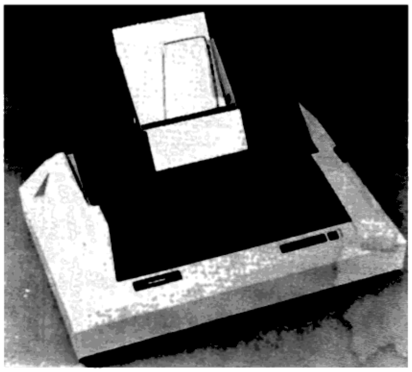
Snapp II Extended Basic

Snapp II Extended Basic is written in machine language to increase execution speed. Extensions are fully integrated into Model II Basic and require no user memory or disk space.

Snapp II includes a cross-reference ability, the ability to display and print program variables, a program line renumbering ability, a cross reference facility for key words or character strings, and the ability to compress a program to the absolute minimum.

It is available for the Model II for $200 and the Model III for $125, from Snappware Inc., 3719 Mantell, Cincinnati, OH 45236, (800) 543-4628.
Reader Service ✓185

The Boss



The Anadex Dual Mode WP-6000

The Anadex Dual Mode WP-6000 Printer

The W.P-6000 dot-matrix serial printer from Anadex provides high speed operation and is capable of 150 characters per second in the letter-quality mode and up to 500 cps in the data processing mode.

This printer features an 18-needle, dot-matrix print head consisting of two vertical rows of nine needles slightly offset from each other in the vertical dimension. Graphics capability is also provided. The WP-6000 can print a variety of foreign fonts and features bi-directional, logic-seeking operation, standard interfaces of RS-232, current loop or parallel (Centronics), and friction feed with an optional adjustable tractor feed.

The WP-6000 costs under $1,800 and is available from Anadex, Inc., 9825 DeSoto Ave., Chatsworth, CA 91331.
Reader Service ✓346

TurboDOS

TurboDOS is a new disk operating system for the Model II, compatible with Digital Research's CP/M, version 2.x.

TurboDOS features file management facilities, a sophisticated buffer manager, a reentrant file manager, automatic print spooling, a command language interpreter, and a command file processor. Other capabilities include an extensive set of utility programs, system date and time functions, standard communications channel interface, and more.

A special introductory price of $195 for the spooling version is being offered from Data-RX, Inc., 686 Lighthouse Ave., Montery, CA 93940, (408) 375-2775.
Reader Service ✓342

Mailing List System

Precision Prototypes has announced an improved TRS-80 (Model I or III) mailing list system to use with large lists. Features include maintenance of all disks in continuous alphabetic or zip order, high speed sort and disk retrieval, and up to 4640 (for Model III) addresses on-line. Two 80-track drives are required.

The system is specially geared to use advertising statistics in selecting and purging names for print outs.

Hardware requirements are 32K, two disk drives, and a printer. The system is available for $69.95 from Precision Prototypes, 410 E. Roca, Refugio, TX 78377. A documentation manual is available separately for $3.95.
Reader Service ✓344

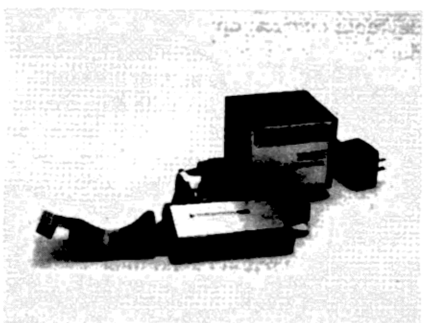
Card Reader Interface

Chatsworth Data Corporation has developed a special interface to the TRS-80 Model III for its MR 500 and OMR 500 card readers. The interface plugs into the I/O bus jack of the Model III.

A software driver and interface that enables the user to input data is supplied with their reader. The MR 500 reader utilizes an electric current technique for reading soft pencil marks; the OMR 500 is an opticle reader.

Educational applications for the TRS-80 include grading tests, attendance and grade reporting, and teaching programming. Business applications for the reader include inventory, labor distribution and time card recording.

For more information on this product contact Chatsworth Data Corporation, 20710 Lassen St., Chatsworth, CA 91311.
Reader Service ✓164



The Chatsworth Card Reader Interface

The Ramlok Memory Protection System

Memory Protection System

Ramlok is a computer equipment protection system that conditions supply line voltage and supplies a safe operating environment for electronic equipment, especially computer memory circuits.

Ramlok filters line voltage with a dual T section RFI filter. It ensures correct voltage limits and provides proper filtering to suppress RFI line noise and an uninterruptable power supply capability. The user provides the battery/inverter, Ramlok provides regulation control, filtering, battery charger, failure indication and multiple circuit and the power distribution panel for equipment connection.

It is sold for $495, from Ladco Development Co., Inc., P.O. Box 464, Olean, NY 14760, (716) 372-0168.
Reader Service ✓160

The RAM Communication Area

The RAM Communication Area is a booklet which is intended for the TRS-80 computer owner.

It is a concise description of the Level II memory locations from 16384-17128 (4000-42E8). The booklet also contains an article dealing with number conversions.

RCA is priced at $4.50 and is available from ABS Suppliers, P.O. Box 8297, Ann Arbor, MI 48107, (313) 971-1404.
Reader Service ✓335

Loan Amortization Schedule

This program allows the user to print a schedule of loan repayments along with

monthly and yearly principal and interest data. One of the options automatically adjusts the pay date away from weekends and holidays. A built-in calendar ensures accuracy to the year 2099.

The program is available for $19.95 on tape for the TRS-80 Models I, II, and III. Contact Precision Prototypes, 410 E. Roca, Refugio, TX 78377.
Reader Service ✓347

Color Computer Newsletter

Rainbow is a newsletter devoted solely to owners of the Color Computer and will be available every month.

A typical issue of *Rainbow* will contain feature stories, hints and tips on operation, sample programs and reviews of Color Computer-associated hardware and software.

Annual subscriptions sell for $12 and can be ordered through *Rainbow*, 5803 Timber Ridge Drive, Prospect, KY 40059.
Reader Service ✓328

Football Scouting

The Football Scouting Report is a system of programs that analyze running, passing, and kicking plays. They run on a Model I or a Model III with 32K, one disk drive, and a printer.

A team can be scouted up to five times before running a composite analysis, or each game can be analyzed separately. The package consists of eight programs that are suitable for both college and high school use. They are available from Precision Prototypes, 410 E. Roca, Refugio, TX 78377 for $89.95.
Reader Service ✓340

Isaac Newton

Isaac Newton is the name of an educational game which challenges players to assemble evidence in support of particular laws of nature.

It is an inductive game allowing players to intervene actively in determining if new information supplied by the computer conforms to the laws of nature in question.

Isaac Newton is available for the TRS-80 for $24.95 from Krell Software, 21 Millbrook Dr., Stony Brook, NY 11790, (516) 751-5139.
Reader Service ✓330

K-8 Math Cross-Reference

K-8 Math Cross Reference

Radio Shack is publishing the *K-8 Math Cross Reference* to be used in conjunction with their computer-based educational programs for kindergarten through eighth grade.

This teachers' reference coordinates the lessons in the K-8 math program for the Model I and Model III to materials in six of the most commonly used elementary math textbooks.

The price of this reference manual is $4.95 from Radio Shack, 1800 One Tandy Center, Fort Worth, TX 76102.
Reader Service ✓181

300 Baud Modem

Kesa Company has introduced the DataSpeak O/A-300, a compact, low cost, 300 baud modem that connects directly to the telephone line with FCC part 68.

The unit, featuring orignate and answer modes, RS232 interface, and test mode, measures 3.5 inch by 4.5 inch by 1.5 inch. A unique "off hook" light alerts the user the phone line is in use. Power is supplied by a wall mounted power converter. Connection to the phone line and the user's telephone is by standard modular RJ-11C jacks.

New phase-lock loop LSI circuitry allows the DataSpeak modem to deliver such performance at the price of $129. For more information contact Kesa Company, 774 San Miguel Ave, Sunnyvale, CA 94086, (408) 746-2738.
Reader Service ✓163


Data Speak

The Documenter

The Documenter is a new programming utility that will flowchart a Basic program and provide the user with a branch map. The Documenter will run on the Model I and the Model III.

This utility is an aid in error detection and will flowchart any program written in standard Radio Shack Basic Level II or Disk Basic. The Documenter is available in 16K, 32K, or 48K tape versions for $19.95 or a 48K disk version for $29.95. For additional information contact P80NUT Software, P.O. Box 490, Lilburn, GA 30247, (404) 469-0056.
Reader Service ✓334

Supersnappx

A fast in-memory sort routine which offers additions to the Model II Basic Interpreter is available from Snappware.

Supersnappx is a collection of expansions to Basic giving added capabilities to Model II. These capabilities include three kinds of PEEKs, three kinds of POKEs, an improved data retrieval function, an elapsed time calculator, a function that states how many file blocks have been used, and more.

This product is available for $100 from Snappware, 3719 Mantell, Cincinnati, OH 45236, (800) 543-4628.
Reader Service ✓327

Game Played to NFL Rules

Super Micro Pro Football is a game played according to NFL rules. It includes a wide range of fast, animated graphics. There are 16 offensive and seven defensive formations from which to choose. As

NEW PRODUCTS

the game progresses, the computer's team adjusts its offensive and defensive strategy according to the game situation. Cumulative game statistics are updated after each play.

Versions are available for the Model I or III, on tape or disk, 16K and up. Prices start at $12.95. For more information contact Micro Pro Systems, Rte. 2, Box 533, Cumming, GA 30130, (404) 887-6814.
Reader Service ✓165

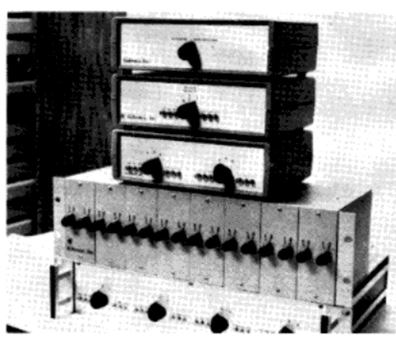
"ABC" Switching Units

Giltronix has introduced a new group of "ABC" switching units which allow the sharing of a common device such as a printer or terminal between two computers.

The units include the GRS-232-S8AB ($99) and the GRS-232-S24AB ($106). The -S8AB switches the eight lines of the RS232 interface pins and the -S24AB switches all 24 lines of the RS232 interface.

The units can be ordered with these options: monitoring capability, dual units, and a rack mounting option.

For more information contact Giltronix Inc., Microcomputer Systems and Services, 450 San Antonio Rd., Suite 44, Palo Alto, CA 94306, (415) 493-1300.
Reader Service ✓168



The "ABC" Switching Units

Design Grids

A full line of computer forms-design grids are available from The House of Grids. These grids are helpful to forms designers and computer programmers to check the accuracy of form proofs.

The grids have an overall size of 12-inches by 18-inches and have the following features: numbered print positions, split print position indicators, and standard size indicators for depth of forms.

The grids are available on stable based film for $40 each or on a paper pad of 50 for $9.50 from The House of Grids, 135 E. York St., Akron, OH 44310, (216) 376-3974.
Reader Service ✓180

Fast Sort

Fast Sort is a short Basic program with embedded machine code. When merged by disk with a user's own TRS-80, Model I or Model III Basic program, it will allow sorts of strings (alphabetizing), integers, and single or double-precision numbers. Ascending or descending order is also selectable.

Typical sort times are 50 seconds for 5000 integers and eight seconds for 1000 double precision numbers. For more information contact Precision Prototypes, 410 E. Roca, Refugio, TX 78377, (512) 526-4758. Fast Sort sells for $19.95.
Reader Service ✓343

Basic Compiler

Radio Shack is offering a Basic compiler for the Model I and Model III computers with 48K and dual disk. This system is a business-oriented Basic intended to be used as a development package for new software.

RSBasic includes three programs: RSBasic compiles programs; BEDIT is the editor for Basic source programs; and RUNBASIC executes compiled Basic programs. All three are provided on 5 1/4-inch floppy disks.

For more information contact Radio Shack, 1800 One Tandy Center, Fort Worth, TX 76102.
Reader Service ✓166

Adjustable Microcomputer Table

Smith System Manufacturing has designed a new microcomputer table which adapts to both the operator and the computer.

The 16 inch deep keyboard surface adjusts vertically on one inch centers from 24 to 28 inches high. The 16 inch deep monitor surface is fixed at 30 inches high. Both tops have radiused corners and soft edges for safety, and are available in either a 30 or 42 inch width. The table has a cantilever design for easy access.

For more information, contact Smith System Manufacturing, P.O. Box 43515, St. Paul, MN 55164, (612) 636-3560.
Reader Service ✓345

TRS-80™ PROGRAMS
SUPER SOFTWARE AT SUPER PRICES

THE ULTIMATE DISK-OPERATING SYSTEM

LDOS™

A completely documented new generation of operating system for the TRS-80. Far superior to any on the market. It is a totally independent device system, capable of device linking, routing, setting, and filtering. LDOS will support 5" and 8" floppies, single/double density, single/double sided, and up to 80 tracks. Full customer service provided with a toll free 800 number. Excellent documentation in a tab indexed manual (over 250 pages). Call or send for more information today.

Available for Model I or III . . . . only $169.00

LDOS™ is a product of Logical Systems Inc.

ALSO AVAILABLE

EDAS 3.5 Model I and III. . . $79.00
EDAS 5.0 Model II. $179.00

Host Model II was. $199.00
now. $99.00

galactic software ltd.
11520 North Port Washington Road
Mequon, Wisconsin 53092

MAILING SYSTEMS

MAIL/FILE - SERIES II

NEW RELEASE

NEW Series II Mail/File has all of the outstanding features of the original Series I Mail/File, PLUS many additional qualities which again will set standards for this type of system. "Word processor" type input editor, fast sorting and fabulous editing capabilities are a few of its features. Name and ZIP code are under constant sort. Series II is compatable with Series I data files.

Model I system will handle up to 600 names.
Model III system will handle up to 1200 names.
Either system available for only. $159.00
Also available for the Model II. $199.00
Handling up to 2500 names per file

MASS/MAIL SUBSCRIPTION SYSTEM AVAILABLE FOR MODEL II. THE ULTIMATE IN MAILING SYSTEMS

"I think the system will stand the test of time . . . it is probably the most sophisticated system around for the micros today"
Mike Schmidt (EDITOR) 80-US Jan/Feb pg. 94

Galactic's Mass/Mail System was designed with the needs of the large mailing list in mind. This system has capabilities of handling up to 10,500 records, with access times of less than 15 seconds and usual access of about one second. All adds, deletes and edits are instant for the operator and

MAILING SYSTEMS

are then completed later in a "batch mode". The System comes complete with extensive documentation and ongoing support. . . . .

Contact Galactic Software for detailed specifications for your exact needs. Available only for the TRS-80 Model II.

BUSINESS SYSTEMS

INVENTORY MASTER SYSTEM

A program to free your needs, handling up to 2700 items. With today's market, keeping on hand only what your demand calls for is reason enough to let your TRS-80 accurately and effectively take care of your inventory. Always know what to stock and when to stock it. Have a multitude of reports at your finger tips. This system has many features which were modeled after a main frame system of large capacity. More information available upon request.

Model I version just. $159.00
Model III version just. $259.00

ALSO AVAILABLE . . .
For the MODEL I or III

STOCK MARKET MONITOR

Cassette version. $89.00
Disk version. $99.00

ULTRA-TREK GAME

Can you master this trip through the galaxies? As commander, will you be able to keep the Enterprise and its crew en route and on patrol? Can you destroy the hostile base? Order today.

Model I or III version only. $19.95

Send me the following:

☐ LDOS Model I	$169.00	☐ INVENTORY MASTER Model I	$159.00
☐ LDOS Model III	$169.00	☐ INVENTORY MASTER Model III	$259.00
☐ EDAS 3.5 Model I	$79.00	☐ STOCK MARKET MONITOR Model I	
☐ EDAS 3.5 Model III	$79.00	cassette	$89.00
☐ EDAS 5.0 Model II	$179.00	disk	$99.00
☐ HOST Model II	$99.00	☐ STOCK MARKET MONITOR Model III	
☐ MAIL/FILE II Model I	$159.00	cassette	$89.00
☐ MAIL/FILE II Model III	$159.00	disk	$99.00
☐ MAIL/FILE II Model II	$199.00	☐ ULTRA TREK Model I	$19.95
☐ INFO ON MASS/MAIL MODEL II		☐ ULTRA TREK Model III	$19.95

☐ I WOULD LIKE MORE INFORMATION ON THE FOLLOWING: ____________________

☐ CHECK ENCLOSED ☐ C.O.D. ☐ VISA CARD NO. ____________
☐ MASTERCARD NO. ____________

NAME ____________________

ADDRESS ____________________

CITY ____________ STATE ________ ZIP ________

Mail to: galactic software ltd. 11520 N. Port Washington Rd. Mequon, WI 53092 J81A

Can't wait? call
414/241-8030

TRS-80 A Trademark of Tandy Corp.

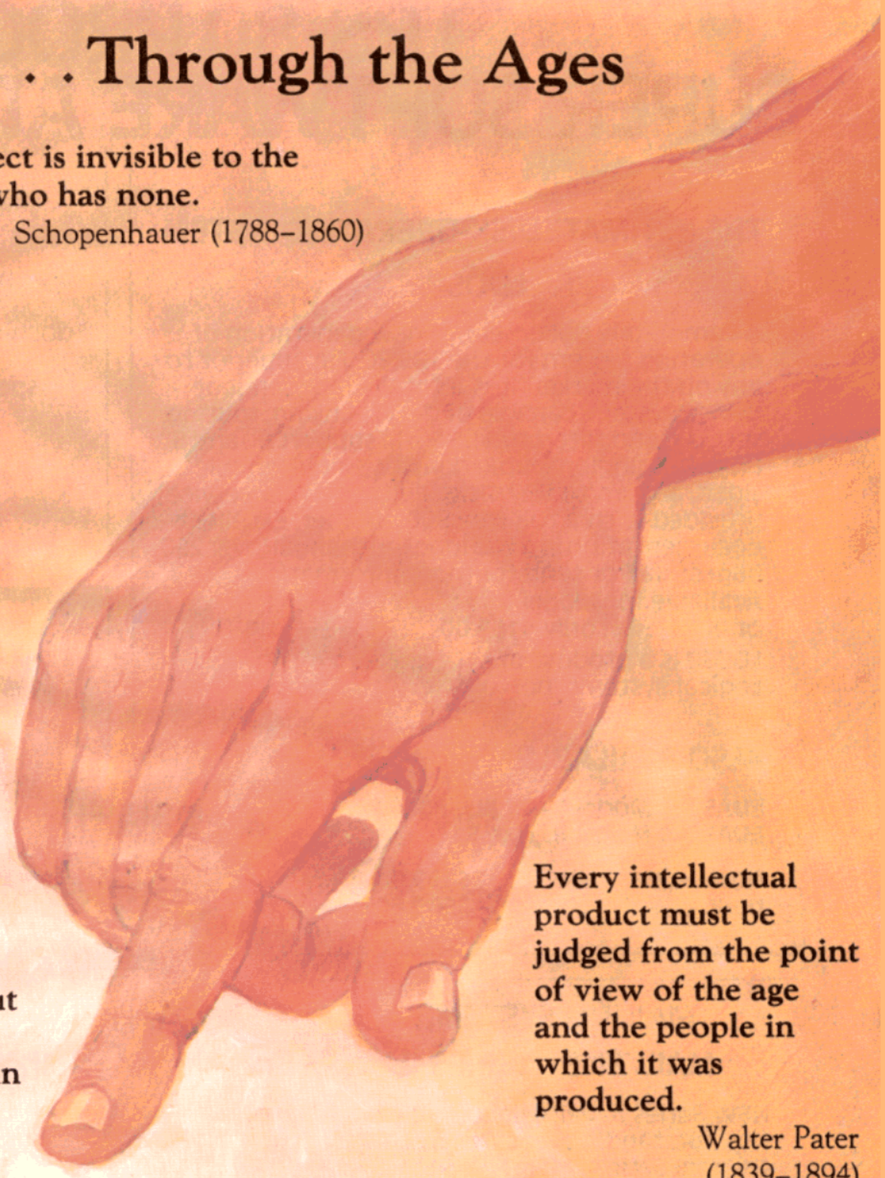
Intelligence . . . Through the Ages

The splendid achievements of the intellect, like the soul, are everlasting.

Sallust (86–34 B.C.)

Intellect is invisible to the man who has none.

Schopenhauer (1788–1860)



The controlling intelligence understands its own nature, and what it does, and whereon it works.

Marcus Aurelius (A.D. 121–180)

The intellectual world is divided into two classes—dilettantes, on the one hand, and pedants on the other.

Miguel De Unamuno (1864–1936)

Thought is the labor of the intellect, reverie is its pleasure.

Victor Hugo (1802–1885)

Every intellectual product must be judged from the point of view of the age and the people in which it was produced.

Walter Pater (1839–1894)

The voice of the intellect is a soft one, but it does not rest until it has gained a hearing . . . This is one of the few points in which one may be optimistic about the future of mankind.

Sigmund Freud (1856–1939)

humanity i love you because when you're hard up you pawn your intelligence to buy a drink.

e. e. cummings (1894–1962)

A test of first rate intelligence is the ability to hold two opposed ideas in the mind at the same time, and still retain the ability to function.

F. Scott Fitzgerald (1896–1940)

The fact that a man knows right from wrong proves his intellectual superiority to the other creatures; but the fact that he can do wrong proves his moral inferiority to any creature that cannot.

Mark Twain (1835–1910)

We should not pretend to understand the world only by the intellect; we apprehend it just as much by feeling.

Carl Gustav Jung (1875–1961)

Intelligence . . . is the faculty of making artificial objects, especially tools, to make tools.

Henri Bergson (1859–1941)

Artificial Intelligence—Technology and The Search for Self

by Chris Brown
Technical Editor

In the early 1950s, two seemingly unrelated events took place. The first was the airing of an episode of the popular television series, "I Love Lucy," in which the lovable redhead and her zany companion Ethel found themselves working on a high speed bon-bon production line. The second was the publication of a scientific paper by Alan Turing titled "Computing Machinery and Intelligence." Worlds apart in audience and approach, these events were, none-the-less, related. Turing's paper posed, for the first time, the philosophical question—"can machines think?" The "I Love Lucy" episode illustrated the essence of intelligence and thinking that Turing was forcing the scientific community to examine.

Turing's thesis, that machines could indeed think, quickly drew fire from philosophers who eagerly mired his theory in arguments over semantics. What, they demanded, do you mean by "thinking?" What is a machine? How is intelligence defined? Though they posed questions worthy of consideration, the philosopher's linguistic nitpicking did nothing to further the cause of scientific inquiry. It did alienate the early Artificial Intelligence community to the point of inaudibility, however.

Semantics were of no concern to Lucy and Ethel. All they knew was that in their immediate environment that damn conveyor belt was supplying bon-bons faster than they could pack them into boxes. So, in a desperate attempt to keep their jobs, they began to eat every tenth bon-bon. Then, as the belt speeded up, they ate every fifth, every third and eventually, every other. Finally, with cheeks bulging like October chipmunks, they ate every bon-bon that came down that line. Though feeling increasingly queasy, they had done what no machine had yet been able to do. They had adjusted their behavior to their surroundings in accordance with the demands of changing conditions. They had adapted and in doing so displayed the intelligence that Alan Turing believed computers would one day be capable of.

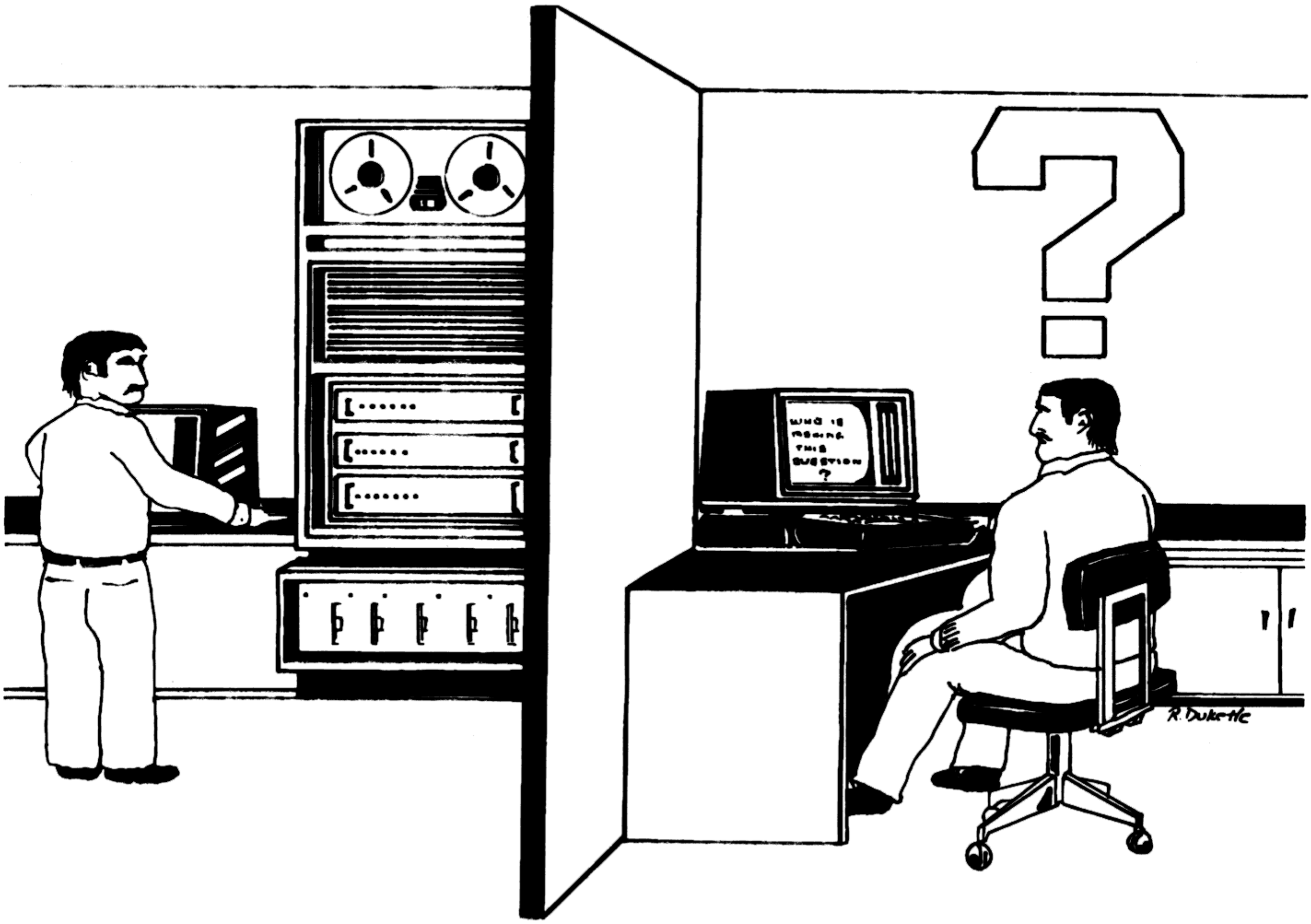
"Ask those involved to define what they are researching. . . You'll get as many definitions as you have researchers."

Technologically, we have come a long way since the days of "I Love Lucy" and Alan Turing. We still don't have computers that perform as well as Lucy and Ethel, however. Philosophers and artificial intelligence researchers are still arguing over semantics and methodology. And most important, Alan Turing's question, "can machines think?" has yet to receive a satisfactory answer.

What It Is

What exactly is artificial intelligence (AI)? In essence it is a method by which we can learn about the nature of knowledge and the nature of man. Ironically, the most functional definition of AI lies with the man who originally broached the issue, Alan Turing. His Turing Test is not concerned with the methods used to produce intelligence but, rather, with the end results. In a modern day version of the Turing Test a human subject is presented with a keyboard and a video monitor which he uses to converse with two unknown sources of information. One of these sources is a machine, the other a human. Though the subject is aware he is dealing with one man and one machine, he does not know which is which. By interrogating his sources, the subject must determine which is the machine. If he fails to do so better than 51 percent of the time, the machine can be said to have successfully simulated human intelligence. So far, no machines have passed this acid test. Increasing numbers are coming close, however.

Ask the people involved in the research to define what exactly it is they are researching and you will get as many definitions as you have researchers. The fact is that AI is a vague field that encompasses the least understood aspects of human existence, conscious and unconscious thought. Patrick Winston, an AI veteran and member of M.I.T.'s artificial intelligence group, defines artificial intelligence research as "the study of ideas which enable computers to do the things



that make people seem intelligent." Philip Jackson, a textbook author and AI researcher for Xerox offers, "AI is the ability of machines to do the things that people would say requires intelligence." Abe Lockman, an AI researcher and professor of computer science at Rutgers University says, "AI involves the creation of systems that simulate intelligent human functions and that handle I/O in a manner similar to the way humans do." The salient feature of each of these definitions is the simulation of intelligent human thought by a machine.

The Goals of AI

It is possible to get a clearer idea of what AI is about by examining its goals. In a succinct statement of purpose, M.I.T.'s Patrick Winston identifies AI's goals as follows: "The central goals of AI are to make computers more useful and to understand the principles which make intelligence possible." The first goal, making computers more useful, is the easier of the two to attain and many of the techniques developed during the course of AI research are routinely used in both programming and hardware design today. Some of these include the structures for languages like Lisp and Logo and the methods of expert systems analysis.

The second goal of artificial intelligence, understanding the principles of intelligence, is not so easily achieved and has much broader ramifications. In AI's continuing search for a mental model that closely resembles the human thought process, many theories have come and gone. At first it was believed that the study of problem solving, pattern recognition and theorem proving would yield the most accurate model of human thinking. Over the past ten years, however, the emphasis has shifted away from these areas. Today, many researchers feel that the key to unlocking the secrets of human cognition lies in discovering how we process language. In university laboratories across the country projects are underway to determine just that.

Programs That Read

At Yale, computer programs with names like Sam (Script Applier Mechanism), Frump (Fast Reading and Understanding Memory Program), Pam (Plan Applier Mechanism) and Politics have been developed in efforts to validate theories about language processing.

Sam is a program that understands stories that have been written according to scripts (data structures that describe situations in certain ways). Sam is capable of creating its own concepts of what has happened in a story by parsing sentences and drawing inferences. It can provide paraphrased versions of stories that are longer than the original by elaborating on what it has learned from its reading.

Frump is a similar but more flexible program. Frump skims a newspaper from beginning to end looking for subjects in which it is interested. Once it finds a relevant subject, it applies the rules of a grammar to determine the information it wants to know. Frump is a fast program by AI standards and its paraphrased summaries read like the copy of newspaper night shift rewriters. In addition, Frump is virtually language independent. It is easily able to

Strictly Business

If you've ever missed a deadline, forgotten an appointment, waited on late deliveries or lost track of projects, put ANGEL to work for you.

ANGEL is the new software program for microcomputers that helps organize and manage your business flow. It's strictly business.

FAST AND EASY

With a series of simple one-line entries, ANGEL can schedule every phase of your projects and appointments, however simple or complex. Then, each day, ANGEL will give you a run-down of every action that needs to be taken to complete all your work on time, whether it's due tomorrow or ten years from now.

ANGEL's large capacity allows you to log and track up to 2,000 individual projects at once. With ANGEL's help, you can eliminate reams of files and hours upon hours of costly business delays and missed deadlines.

SPEAKS SIMPLE ENGLISH

ANGEL speaks plain English. No computer language to memorize, no codes to decipher. Sit down and talk to it, face-to-face. If ANGEL doesn't understand, it will respond in plain English. If you make a program error, ANGEL tells you how to correct your mistake through the use of a built-in video manual. ANGEL is so easy to use, you can turn it on and enter 30 items in less than five minutes.

WORKS FOR ANY BUSINESS

You can custom-program ANGEL for any business operation — from oil exploration to insurance, home construction to retail sales.

With a memory capacity that's limited only by the capacity of your disk storage, ANGEL tracks multiple projects for busy executives. The disk comes with an easy-to-understand printed manual, to back up the built-in set of instructions of the program itself.

In addition, a data-proof safeguard system will not let you accidentally erase entries, and a special password code of your choice allows you to keep confidential files.

Put ANGEL to work for you today. You'll never lose track of your work flow again.

ANGEL is recommended for use with any disk-based system that has 48k memory or more. It will operate with any CP/M*-based system and the TRS 80** (Model I, II or III). The program comes on standard 5¼-or 8-inch disk, with storage sleeve. Now only **$295.**

ANGEL is available only through Time Management Software. It is not sold through any outlets at any price. Order the complete program for better business by calling one of our toll-free numbers or by returning the order blank below.

Call now toll-free to order (Ask for operator 602)

1-800-824-7888 (nationwide, except California, Alaska and Hawaii)

1-800-852-7777 (in California)

1-800-824-7919 (in Alaska and Hawaii)

VISA MasterCard

Yes . . . I need ANGEL to work for me, to organize my business with ultimate ease and efficiency.

NAME ____________________

ADDRESS ____________________

CITY __________ STATE __________ ZIP __________

I have enclosed $295 to avoid delays, shipping and handling charges. 10-D1

☐ Check ☐ Money Order

☐ C.O.D. (If delivery address is different than above, please specify)

____

Please specify one: ☐ 5¼-inch disk ☐ 8-inch disk

Please specify one: ☐ CP/M* Operating System ☐ TRSDOS** Operating System

Price includes tax, handling and shipping (except C.O.D. orders)

TIME MANAGEMENT SOFTWARE™

123 E. Broadway
P.O. Box 727
Cushing, Oklahoma 74023

* Registered trademark of Digital Research, Inc.
** Registered trademark of Tandy Corp.

> *"What do you mean by "thinking?" What is a machine? How is intelligence defined? The philosopher's nitpicking did nothing . . ."*

paraphrase in English, Spanish and Russian, depending on which grammar is in its database.

Hawks and Doves

Politics is a program that uses a database of political philosophy to answer hypothetical questions regarding the motives behind (and consequences of) certain political actions. It accepts input in the form of questions, examines its data base and supplies answers in tersely worded sentences that would be much too concise for most State Department spokesmen. When loaded with its conservative/hawk database and asked why the Russians had massed troops on the Czechoslovakian frontier, the program answered, "Because Russia thinks it can take control by sending troops in." When asked what Russia would do next, it answered, "Order its troops in." When asked what the U.S. could do about the situation, it suggested, "Intervene militarily." Loading in the liberal/dove database would have resulted in different answers. Politics is used to supply scenarios of hypothetical situations according to the tenets of various political philosophies. At present, programs similar to Politics are helping decision makers examine the consequences of their actions before things have gone too far.

In all these programs, researchers have been able to get machines to interpret language and draw intelligent conclusions. In fields like medicine and geology, programs such as these have already performed at the level of human specialists and experts. Whether helping to diagnose bacteriologic infections from a myriad of symptoms (MYCIN) or predicting the location of natural resources from existing geologic conditions (Prospector), the value of such programs to society is obvious. This is the practical aspect of the often esoteric field of artificial intelligence. On a larger scale, AI can help philosophers, psychologists and linguists in their work by allowing the integrated thought processes each of these people study to be dissected, examined and mimicked by machines.

Conservative Outlook

For Abe Lockman and other researchers, AI is a promise of the future. Unfortunately, over-zealous members of the AI community have damaged their own credibility by making promises they have not yet been able to keep. Today, most AI people are more conservative in outlook than they were ten years ago. They prefer to emphasize the small, but real, gains being made rather than speculate on what might be.

Lockman told *80 Microcomputing*, "Many grandiose claims have been made about what AI has done. In my opinion, we have not come very far. Sure, we have programs that sit up and bark in toy domains, but the real problem is how generalizable are these theories of intelligence?" In his AI-oriented computer department at Rutgers, Lockman's 700 students, many of whom are interested in careers in artificial intelligence, are being taught the principles of natural language processing.

Lockman says, "In my research I'm looking for the mechanism that allows a human to instantly get full understanding of a text. I believe that all of our knowledge of the world is brought to bear on each word we read. I want to know how that knowledge is organized and, especially, how it is accessed by our brains." Lockman admits that he is a long way from the answers to these questions. In fact, he hasn't yet used the computer to validate his theories. He candidly admits, "I haven't programmed my theories yet because, unfortunately, I'm able to disprove most of them on paper."

When asked about the future of AI, Lockman was skeptical. "I see no major breakthroughs on the horizon. We will continue to build our limited systems and basically do more of the same." On a more optimistic note Lockman indicated that some progress will occur due to the increasing cooperation between the traditionally warring camps of AI theorists, philosophers, linguists and psychologists, all of whom are taking an increased interest in each other's work.

Turning To Science

The most intriguing thing about AI research is that it is an investigation of self: how we think, why we interpret as we do, and ultimately, who we are. Man has been perplexed by his existence since earliest history and has used myth, religion and magic to explain his existential dilemma. In our post-industrial, high-technology age we are increasingly turning to science for answers to the timeless questions of existence. By providing us with a clearer understanding of the human mental process, artificial intelligence research may eventually lead us to a better understanding of self. ■

Reader Service for facing page ✓567→

Once man's world was the center of the universe.
Then came Copernican astronomy...
Once man was a being apart from the rest of creation.
Then came Darwin and evolution...
Once man understood his own behavior.
Then came Freudian psychoanalysis...

And Now Artificial Intelligence

Harold Nelson
Kilobaud Microcomputing *Technical Editor*

There have always been those who view such developments as blows to man's humanity, his uniqueness and superiority. But, by and large, such developments in man's understanding of himself have been irresistible, beyond suppression. We seem to yearn to more fully understand what we are. Artificial Intelligence (AI) may be the latest field of research to give us new tools to extend that understanding. Getting a feel for what AI is all about could lead to some astonishing insights.

Roast Mules

Can you rearrange the letters of roast mules to form one fairly common English word? (It's something most first graders can do.)

Once you solve that puzzle, can you write a program that can reproduce what you have done? Can you write a program that can solve the puzzle?

It might seem that once we have solved it, writing the program should be easy. In principle (which is something quite different from practice), it might be.

One way to solve a puzzle like this would be to list all permutations of the letters, then pick out the word (I think there is only one). Even if we started to seriously consider this method, the fact that there are 3,628,800 such permutations would soon cause us to abandon it. We would be more likely to start arranging the letters into familiar groupings—syllables—then rearrange these until we found the word. That still is not as easy as it sounds. But when we do this something else is occurring. There are quite a number of combinations (such as oaue and ssrtlm) that we just do not consider—that we reject "without even thinking about it." Certain combinations (i.e., sounds) do not occur in our language, so we don't even think about them. Others do, and we work with those. What usually happens is that we chance upon one or two of the proper syllables and the solution seems to jump out at us.

How do we write a program that can do that? What is it that we do that makes the solution seem to jump out once we get close? How then do we write a program capable of doing something we do when we're not even sure how we do it? No one said the program had to get the solution in the very same way we did. And, of course, if we could get a program to solve the puzzle, it might help us to understand more fully what it was we did when we solved it.

If all that's a little confusing, it may be because we have quickly, albeit via a trivial example, arrived at the kinds of issues that are at the heart of much of the current AI research. Let's take a look at some products of AI research—some programs that, in the words of MIT's Marvin Minsky, "do things that would require intelligence if done by men." Actually, some of them only seem to require intelligence at first.

Eliza

This "psychiatric interviewer" program (some versions are called Doctor), written by Joseph Weizenbaum, has been around for over 15 years. It is worth a little consideration, if for no other reason, because it has "fooled" some people. After an encounter with the program (difficult as it may be to believe), some people who had a sufficient willing suspension of disbelief or naivete were convinced the program had really understood them.

If you "play along", you can produce a fairly realistic dialogue with the program. For example (the program's lines are in italics):

Hello. Tell me your problem.
I don't want to.
Why do you suppose you don't want to?

the supreme command

We're Offering You Our Company Secret.
It's a brand new American-designed program generator that can eliminate the need to spend thousands of dollars on specialty programs.

And, It's Available Today.
It's the microcomputer program that Time Management Software is using to write its own professional software packages. And now, you can use it to write all the programs you'll ever need. We call it *GENESIS*. You'll call it incredible, because we have yet to find a program on the market that GENESIS can't create.

GENESIS is the next generation. Designed by a team of doctoral-level programmers, GENESIS runs on all CP/M* 2.XX systems, using compiled PL/1-80*.

The code generated by GENESIS is fast and efficient, and this new product's capacity to code difficult algorithms is virtually unlimited.

The speed of code generation itself is literally awesome. Preliminary tests have indicated an ability to generate efficient code at a rate well in excess of four lines per second. Clearly, GENESIS can write months of code in a matter of minutes.

With You As Architect, Genesis Is The Ultimate Master Builder.
Word processing, accounting procedures, statistical analysis, time management studies . . . GENESIS can write any functions you could possibly need. Naturally, complete program changes are quickly and easily made.

You speak plain American English to GENESIS — no code, no difficult introductory language to memorize. The only time you spend with GENESIS is in deciding the specifications of your program.

Simply Stated, Genesis Is The Best Microcomputer Program Generator In The World. Period.
Genesis comes with on-line documentation, as well as a complete concise printed manual, written by a professional manual writer.

Due to its sophistication, we recommend this program for use with any disk system with 48k or more, using CP/M* 2.XX. The code which is generated is CBASIC*.

(Note: Even though GENESIS runs in compiled PL/1-80* it is not necessary to have PL/1-80* in order to run the program.)

The program comes on standard 5¼-or 8-inch disk, with storage sleeve. The program that will allow maximum use of your hardware . . . an introductory price of **$500.**

GENESIS is available now directly from Time Management Software, and is not sold through any other outlet. Time Management Software will send free updates of new material and products to GENESIS owners as they develop, to make sure you've got everything you need.

Order the most advanced program generator system on the market today, by calling one of the toll-free numbers or by returning the order blank below.

Call now toll-free to order. (Ask for operator 603.)

1-800-824-7888 (Nationwide, except California, Alaska and Hawaii)
1-800-852-7777 (In California)
1-800-824-7919 (In Alaska and Hawaii)

VISA MasterCard

YES . . . I need GENESIS to develop the ultimate program and to take fullest advantage of my hardware.

NAME ____________________

ADDRESS ____________________

CITY ____________ STATE ____________ ZIP ____________

I have enclosed **$500** to avoid delays, shipping and handling charges.

☐ Check ☐ Money Order ☐ C.O.D. (If delivery address is different than above, please specify.)

____

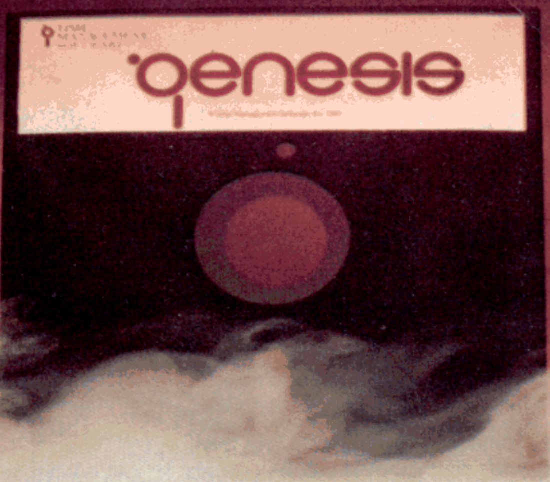
Please specify one: ☐ 5¼-inch disk ☐ 8-inch disk

Price includes tax, handling and shipping (except C.O.D. orders.) 10-D1

TIME MANAGEMENT SOFTWARE™

123 E. Broadway
P.O. Box 727
Cushing, Oklahoma 74023

*Registered Trademarks of Digital Research Inc.



"SHRDLU understands questions and requests put to it in English."

I could never talk to my father.
Tell me more about your family.
None of them liked me.
Perhaps in your fantasies none of them liked you.

Of course, the program is not *really* capable of understanding. Compared to some other programs, it is a very unintelligent program. Eliza achieves its effects by the clever use of a few gimmicks. A more skeptical interviewee quickly reveals the program's weaknesses:

Go away.
I'm not sure I understand you.
I think you're coming apart.
Why do you suppose you think you're coming apart?
You are coming apart.
Perhaps in your fantasies you are coming apart.
No. You are in reality.
Perhaps in your fantasies you are in reality.

This rather inane little exchange reveals Eliza's limited language handling capability.

Animal

A somewhat more interesting program, capable of learning as it is used is called Animal. This is a game in which the program tries to guess the name of the animal chosen by the user. When it makes an incorrect guess, it asks for the name of the user's animal and some information about it. It stores this information and recalls it at appropriate times in subsequent games. In one version the program begins by asking "Does it have horns?" If the answer is yes, it guesses the animal is a buffalo. If the answer is no, it guesses a butterfly. After a few wrong guesses and accumulating information provided by the user, a dialogue with the program might look like this:

Does it have horns?
No.
Does it have a beak?
No.
Does it have four legs?
Yes.
Is it true that it can climb trees?
No.
Does it have a curly tail?
No.
Does it give milk?
Yes.
Is it a cow?
Yes.

Notice that the user-provided information is stored in "branches." The program will not ask about dogs or cows when the exchange is about an animal with horns.

Does it have horns?
Yes.
Does it live in the woods near water?
No.
Is it true that it is smaller than a buffalo?
Yes.
Does it live in high rocky places?
Yes.
Is it an ibex?
Yes.

SHRDLU

One of the most intriguing and significant AI programs was produced by Terry Winograd in the AI Lab at MIT. SHRDLU is capable of understanding questions and requests put to it in English; it is capable of giving English answers to questions and of performing requested tasks even if they require multiple operations. As if that were not enough, the program can understand what it has done and why it has done it, and it can give competent English descriptions of these.

SHRDLU functions in a limited domain called a "blocks world." This consists of a variety of variously-colored blocks, pyramids, boxes, and so forth. The program answers questions about the arrangement of those objects, provided the questions are precisely stated. If they are not, the program asks for a clarification. SHRDLU will perform requested rearrangements of the shapes and explain what intermediate action, not specifically requested, was performed.

Douglas Hofstadter, who provides an absolutely insightful treatment of Winograd's SHRDLU in *Godel, Escher, Bach,* says that while "SHRDLU may not be isomorphic to what we do . . . the act of creating it and thinking about it offers tremendous insight into the way intelligence works."

Internist

The study of "expert systems" is leading some researchers to develop more practical programs. Internist, developed by Myers and Pople at the University of Pittsburgh, reproduces the special diagnostic skills of a doctor of internal medicine. The program collects data on a patient's symptoms, case history, test results, and so forth. Acting much like a human diagnostician, the program then decides which possibilities to delve into, asks specific questions about a possible illness and suggests a diagnosis.

The developers of Internist believe it will be ready for field testing at institutions other than Pittsburgh by 1983.

More Roast Mules, Algorithms and Heuristics

Our discussion of how we might solve the puzzle presented at the beginning of this article exhibited two approaches to programming. The first is the usually straightforward method of writing the code for a given algorithm. In the case of our example, the algorithm might be:

- generate one combination of the 10 letters.
- check this with a standard English dictionary to see if it is a word.
- if it is, stop.
- if not, go back to the first step and start another round.

It wouldn't be hard to write a procedure to generate the permutations, especially in a good AI language like Lisp. It wouldn't even be too difficult (just very boring) to give our program access to the words in a standard dictionary—all we really need would be the listings under A, E, L, M, O, R, S, T and U. Somewhere between the first and the 3,628,800th iteration, we would have our solution.

While this would get the job done, it would be neither elegant nor very satisfying. We might somehow feel uneasy about reproducing in so mechanical a fashion something we do in a very sophisticated manner. We might try to write a more sophisticated procedure, which could include various linguistic principles. Basically we would want to teach the computer some of our language—we would want the program to produce and evaluate syllables just as we do. We would want it to be able to combine these to form trial words just as we do. Using such principles or "rules of thumb" is an heuristic approach to our programming problem.

It is easy to see that the first two programs (Eliza and Animal) employ the straightforward algorithmic approach, while the latter two make use of heuristics. Actually, SHRDLU and Internist probably employ both approaches. Programming heuristics can give us a more sophisticated, more intelligent-appearing program.

Of course, to program in this way implies some understanding of the mental rules of thumb we use when solving our puzzle. This is what Hofstadter is referring to in his statement about SHRDLU.

Are Machines and Programs Intelligent?

AI researchers for the most part are not

". . . one theorem-proving program has correctly proved a theorem with a proof the author was unfamiliar with."

trying to produce mechanical brains. We are a great many years away from any such possibility. In fact, there does not seem to be any hard knowledge on how the brain interprets basic neural activity into "conscious thought." So even if we had the hardware ability readily available to construct a machine that could reproduce what the brain does in this interpretation, the levels of interpretation in the brain and in the machine would not necessarily be isomorphic—there would not have to be a one-to-one correspondence between the levels of interpretation in the brain and those in the machine. While trying to construct such a machine could lead us to new insights about what happens in the brain when thought takes place, we are not on the verge of being replaced by mechanical thinkers.

In spite of the fact that algorithms have been written for activities such as juggling and bike riding, I don't think anyone has tried, in software, to get a TRS-80 to turn a somersault. Still, programs designed to do things that normally require human intelligence have produced some interesting and occasionally surprising results. For example, one theorem-proving program is known to have correctly proved a theorem with a proof the program's author was not familiar with. In a case such as this it is hard to say whether the intelligence exhibited by the program was its own or merely a reflection of its author's intelligence.

Would our puzzle-solving program be intelligent if it solved the puzzle in a way even similar to our way of solving it?

The answer to this question as well as the puzzle itself is, in the over-used last words, left to the reader as an exercise. ■

AI Related Reading

If the topic of artificial intelligence is something you would like to enter more deeply into, here are some interesting and readable books on the subject:

Boden, Margaret, *Artificial Intelligence and Natural Man*, Basic Books, New York. 1977.
Hofstadter, Douglas, *Godel, Escher, Bach*, Basic Books, New York. 1979
Winston, Patrick, *Artificial Intelligence*, Addison-Wesley, Reading, Mass. 1977
Abelson, H. and di Sessa, A. *Turtle Geometry*, The MIT Press, Cambridge, Mass. 1981
Papert, Seymour, *Mindstorms*, Basic Books, New York, 1980.
Ringle, Martin (ed.) *Philosophical Perspectives in Artificial Intelligence*, Humanities Press, New York. 1979
Kent, Ernest, *The Brains of Men and Machines*, Byte/McGraw Hill, New York. 1981

✓See List of Advertisers on page 386

See List of Advertisers on page 386

A discussion of the Pulitzer Prize-winning book.

Godel, Escher, Bach

Godel, Escher, Bach: An Eternal Golden Braid
Basic Books, Inc.
New York, NY
Hardcover, 777 pp.
$20.50

by Nancy Robertson

M. C. Escher's Canon Crabs

Alan Turing, a mathematician and influential pioneer of computer science, wrote, "I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."

Turing made that prediction over 30 years ago. Perhaps in two more decades it will be true. But in our time, the debate over the question, "Can machines think?" is raging and emotional. It has been fanned by the uncanny growth in computer production, and aired in general interest publications such as the *New York Times Sunday Magazine*. Like the theory of evolution, the question threatens the ego of our species. Yet, artificial-intelligence research is undeniably a tool toward understanding human intelligence.

Douglas Hofstader, a young physicist and computer scientist, frames the questions of the artificial intelligence (AI) debate eloquently in *Godel, Escher, Back: An Eternal Golden Braid* (Basic Books, Inc., 1979, New York, NY). In 1980 the book won a Pulitzer. Later that year, it reached the *New York Times* best sellers list.

It is a deep and provocative volume that illustrates the infinite complexities of intelligence. Using the works of Kurt Godel, M.C. Escher and Johann Sebastian Bach, Hofstader identifies three vital characteristics of human intelligence: hierarchies, strange loops, and paradox. They can be interpreted as the three strands of the title's "eternal golden braid." The limited occurrence of these qualities in our most recent and impressive computing achievements reflects on our limited understanding of human intelligence.

Artificial intelligence is not merely advanced computer science, but a composite of studies. "In our century the time was ripe for computers—" the author writes, "computers beyond the wildest dreams of Pascal, Leibniz, Babbage, or Lady Lovelace. In the 1930's and 1940's, the first 'giant electronic brains' were designed and built. They catalyzed the convergence of three previously desperate areas: the theory of axiomatic reasoning, the study of mechanical computation, and the psychology of intelligence."

It seems natural to consider the nature of human intelligence before mechanical

"It is a deep and provocative volume that illustrates the infinite complexities of intelligence."

computation. It is more familiar to us. Yet, it's obvious that a simple dictionary definition of intelligence or intellect is trivial.

Midway through the book, Hofstader writes, "Our confusion about who we are is certainly related to the fact that we consist of a large set of levels, and we use overlapping language to describe ourselves on all of those levels." Hofstader's discussion of those levels, beginning with Bach, Escher and Godel, is one of the books most profound statements.

While the choice of Godel, Escher and Bach at first appears incongruous, Hofstader presents their work as "shadows cast in different directions by some central essence." The opening chapter of the book simultaneously introduces these men and Hofstader's concept of levels of meaning.

The author begins by considering Bach's *Musical Offering*. The title of the composition refers to the fact that it was dedicated, or offered, to Frederick the Great, who was the reigning king of Prussia.

Bach wrote an inscription on the original score, *Regis Iussu Cantio Et Reliqua Canonica Arte Resoluta*. (It translates to "At the King's Command, the Song and the Remainder Resolved with the Canonic Art.") The initials form an acrostic, spelling the Italian word *ricercar*, which meant both "to seek" and "fugue." A canon, which is the musical form of most sections of the *Musical Offering*, is a strict single-theme fugue. There is also a great deal to seek in the *Musical Offering*. In this short example, Hofstader introduces the potential for multiple layers of meaning.

"The idea of a canon," Hofstader explains," is that one single theme is played against itself. This is done by having 'copies' of the theme played by the various participating voices." Fugues follow the same principles of composition, but are based on one or more themes. These themes may be varied in time or pitch. They may be inverted or reversed. For instance, what musicians commonly call a "crab canon" begins with a theme that is played backwards in its second and final rendition.

Hofstader uses the term "isomorphic" to describe the relationship of the variations to the original theme. In other words, the variations are altered forms of the original theme. It is possible to extrapolate a theme from any of its variations. The role of isomorphism in thought processes is stressed again and again throughout the book.

Hofstader wants us to recognize that the form or structure of our thoughts have implicit meaning. The structure of the thoughts we iterate reveals something about the sublevel thoughts that build expression.

In the *Musical Offering* the *Canon per Tonos* has a significant structure. In each expression of the theme, Bach modulates the key upward. For instance, the first statement of the melody and accompanying voices begins in C minor, but the resolution is in D minor. Normally, variations end or "resolve" in the same key that they begin. Opening in D minor, the second statement of the *Canon per Tonos* theme resolves in yet a higher key. The pattern continues to spiral through six modulations. Bach has broken down our normal expectations, yet surprisingly brings the piece back to its original key in the final modulation.

A canon or fugue is clearly a hierarchical system. The rules of composition and the themes to which they are applied produce distinct patterns and levels within the structure.

The *Canon per Tonos* takes us a step farther into Hofstader's theory. This particular canon is an example of a strange loop. "The 'Strange Loop' phenomenon occurs whenever, by moving upwards (or downwards) through the levels of some hierarchical system, we unexpectedly find ourselves right back where we started."

This concept leads us smoothly to M.C. Escher, a Twentieth-century lithographer and print master who represented his fascination with strange loops visually. Escher is known for prints such as *Waterfall* and *Drawing Hands*. In *Waterfall*, water travels upward through a series of three sluices, drops from a waterfall, and begins its defiance of gravity all over again in the same sluiceway. *Drawing Hands* depicts a sheet of paper on a drawing board. The left hand, which is drawn on the paper, is drawing a right hand, which is drawing the left hand.

"The viewer cannot help getting caught up in Escher's implied chain of levels, in which, for any one level, there is always another level above it of greater 'reality,' and likewise, there is always a level below, 'more imaginary' than it is," Hofstader writes. In *Drawing Hands* there is a level of "greater reality" just beyond the print: Escher's hand has drawn the hands that

Eta Oin: Can a pyramid support a pyramid?
SHRDLU: I DON'T KNOW.
Dr. Tony Earrwig: The question makes sense, but there is no example in the scene. SHRDLU has no way of looking into the details of its programs, even though these ultimately define its capabilities.
Eta Oin: Stack up two pyramids.
SHRDLU: I CAN'T.
Dr. Tony Earrwig: SHRDLU tries and fails.
Eta Oin: The blue pyramid is mine.
SHRDLU: I UNDERSTAND.
Dr. Tony Earrwig: All of the information about colors, shapes, sizes and locations of the objects is known to SHRDLU initially. Possession is left as an unknown property of objects about which it can be told. It stores this information in its data base.
Eta Oin: I own all blocks which are not red, but I don't own anything which supports a pyramid.
SHRDLU: I UNDERSTAND.
Dr. Tony Earrwig: The result of understanding these sentences is the creation of new procedures within SHRDLU for deducing whether you own something.
Eta Oin: Do I own the box?
SHRDLU: NO.
Dr. Tony Earrwig: The box supports whatever it contains, including a pyramid.
Eta Oin: Do I own anything in the box?
SHRDLU: YES, TWO THINGS: THE BLUE BLOCK AND THE BLUE PYRAMID.

Example 1

"So far in the development of computer science, conscious self-reference, the recognition of paradox, and understanding the concept of infinity are all elusive."

draw each other.

The print itself is characterized as a two-step strange loop: The right hand draws the left; and the left hand draws the right. The strange loop of *Drawing Hands* is represented in Fig. 1.

Hofstader's term "strange loop" is a finite representation of infinity. "In... strange loops... there is a conflict between the finite and the infinite, and hence a strong sense of paradox."

While Hofstader sees Escher's prints as "visual analogues to the canons of Bach," the paradox embedded in them leads to the study of Godel. Kurt Godel was a German mathematician who published a treatise on mathemataical reasoning in 1931. The paper showed "that no axiomatic system whatsoever could produce all number-theoretical truths, unless it were an inconsistent system!"

Using the strict rules of formal mathematical system to generate and prove theorems of number theory, Godel discovered a true theorem that could *not* be proven. The discovery of this paradox lead to his Incompleteness Theorem and treatise. The validity of his argument threatened the foundations of mathematics, which is dedicated to the eradication of incongruity.

Outside of mathematics, paradox has been a recognized characteristic of human thought since ancient times. Epimenides was a native of Crete who is credited with "one immortal statement: 'All Cretans are liars.' " Was Epimenides telling the truth?

The Epimenides paradox, or "liar's paradox," is a one-step strange loop. A two-step strange loop expressing a similar paradox could be:

"The following sentence is false.
The preceding sentence is true."

This two-step loop is closer to the heart of Godel's Theorem.

Called TNT for Typographical Number Theory, Godel constructed a formal mathematical system that could write self-referential statements about natural numbers. Applying self-reference to numbers was brilliant, regardless of the additional discovery of paradox. Self-reference was a recognized characteristic of human dialogue, of literature and art—but not at all characteristic of our traditional perception of mathematics.

In TNT language can be represented by symbols, which in turn can be represented by numbers. For instance, the English language version of the first axiom of TNT is, "For all variables *a* there does not exist a successor of *a* that is equal to zero." Symbolically, this statement can be represented by: a: $Sa = 0$. The numeric sequence which expresses the same axiom is: 666,262,636, 123,262,111,666.

Using TNT to generate and prove theorems of number theory, Godel eventually derived Theorem G. Hofstader roughly translates this to "G is not a theorem of TNT." In a two-step strange loop, it might be represented as follows:

G is a true statement of number theory.
G cannot be proven.

While the Epimenides paradox rests on the fact that the statement is neither true nor false, Godel's Theorem G is true, but defies proof.

The important thing to grasp about Godel's work is that paradox and self-reference are not limited to language, music and art—paradox and self-reference are found in number relations. Both paradox and self-reference are universal characteristics of human thought. Godel's work reinforces our knowledge that numbers, like words, are symbolic (or isomorphic) representations of ideas.

So far in the development of computer science, conscious self-reference, the recognition of paradox, and understanding the concept of infinity are all elusive. Yet computers, like humans, base their thought operations on hierarchical sytems. Between computer hardware—memory registers, CPUs I/O devices and circuitry—and high-level languages such as Lisp there are several functions. There is a machine language, assemblers, assembly code, interpreters, low-level language, etc. Fig. 2 is a diagram, taken from *Godel, Escher, Bach*, representing the hierarchy of intelligence in a computer.

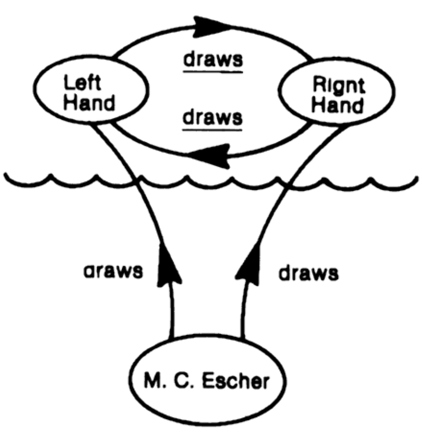


Fig. 1

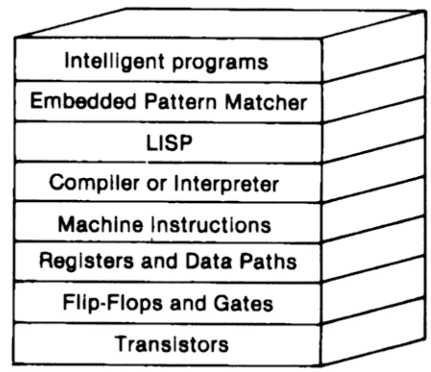


Fig. 2

Hofstader believes that there are human processes that could be delineated that serve the same functions. On the lowest level, he compares hardware and the brain as an organ. For an example of a higher level analogy, take a computer operating system. "It is virtually certain that there are somewhat parallel things which take place in the brain: handling of many stimuli at the same time; decisions of what should have priority over what and for how long; instantaneous interrupts caused by emergencies or other unexpected occurrences..."

Yet computers have seldom demonstrated understanding, self-awareness or creativity. A computer that generates haiku or music from a random mixture of programmed phrases is not creative any more than a calculator is creative. The computer does not intend the resulting lines and melodies to have meaning. It does not understand the explicit definition of the sentence it generates. It does not appreciate the tones of the musical themes it combines. Hofstader writes that:

> An intelligent program would presumably be one which is versatile enough to solve problems of many different sorts. It would learn to do each different one and would accumulate exeprience in doing so. It would be able to work within that set of rules and yet, also, at appropriate moments, to step back and make a judgement about whether working within that set of rules is likely to be profitable in terms of some overall set of goals which it has. It would be able to choose to stop working within a given framework, if need be, and to create a new framework of rules within which to work for a while.

This is the sort of intelligence Godel used

"Yet computers, like humans, base their thought operations on hierarchical systems."

in discovering his Incompleteness Theorem, that Escher used in most of his art, and that Bach exemplified in the *Canon per Tonos*.

Hofstader believes that the current inability of programming and computer science to achieve such program reflects our limited understanding of human thought. He believes "One key for understanding and creation of intelligence lies in the constant development and refinement of the languages."

He argues that Terry Winograd's AI program SHRDLU had demonstrated a degree of understanding in a machine. The program, which was written while Winograd was a graduate student at MIT, converses with humans to manipulate imaginary blocks. Hofstader writes that the program was designed to:

- Understand questions in English about the situation;
- Give answers in English to questions about the situation;
- Understand requests in English to manipulate the blocks;
- Break down each request into a sequence of operations it could do;
- Understand what it had done and for what reasons;
- Describe its actions and their reasons, in English.

Like Hofstader, Winograd emphasizes the importance of the tangled hierarchy of language. He modeled SHRDLU on his view of human thought and human language. Rather than separating the various functions of the program into modules, he intertwined them. The program is so convoluted that other programmers initially thought it was absurd. Yet, the achievements of SHRDLU are impressive.

Example 1 is a segment of a transcribed converstaion between SHRDLU and a visitor. (The words of Dr. Earrwig are actually Winograd's. Eta Oin is a fictional name of a real visitor.) In this short passage, SHRDLU demonstrates an ability to learn concepts.

In an article about SHRDLU Winograd wrote:

> One of the basic viewpoints underlying the model is that all language use can be thought of as a way of activating procedures within the hearer. We can think of any utterance as a program—one that indirectly causes a set of operations to be carried out within the hearer's cognitive system. This "program writing" is indirect in the sense that we are dealing with an intelligent interpreter, who may take a set of actions which are quite different from those the speaker intended. The exact form is determined by his knowledge of the world, his expectations about the person talking to him, etc. In this program we have a simple version of this process of interpretation as it takes place in the robot. Each sentence interpreted by the robot is converted to a set of instructions in Planner (an AI computer language). The program that is created is then executed to achieve the desired effect.

One of Winograd's statements is especially important. "We can think of any utterance as a program—one that indirectly causes a set of operations to be carried out within the hearer's cognitive system." These "programs" and "sets of operations" are the sorts of levels Hofstader wants us to recognize in our own thinking.

Hofstader must stress his point by exam-

Continued on page 192

The 'Eternal Golden Braid'
A Study of Artificial Intelligence

by Nancy Robertson

At first glance, the response to *Godel, Escher, Bach* seems as incongruous as the "eternal golden braid" that weaves the genius of these three men together in a study of artificial intelligence. In 1980 Douglas Hofstader's opus, a book of over 700 pages in Basic Book's hardbound edition, won the Pulitzer Prize for general nonfiction.

Martin Gardner, who had reviewed the book for *Scientific American* wrote, "Every few decades an unknown author brings out a book of such depth, clarity, range, wit, beauty and originality that it is recognized at once as a major literary event." So it was for *Godel, Escher, Bach*.

The manuscript fell into the hands of Martin Kessler, president of Basic Books, New York, NY, when a friend of his who had hoped to publish the volume through a university press despaired of the large task. When the enormous bundle of manuscript and diagrams arrived in the mail, Kessler's first reaction was "a sinking feeling, but then I took it home and began reading it—and I got hooked, at first by the puzzles" Hofstader used to illustrate his points.

Kessler decided to go ahead with publication, although the potential production cost appeared prohibitive. "It looked like we'd have to charge $28 a copy to make our money back. The nagging question was who would buy the book at that price. While interest in the manuscript began to show that the book would probably attract a wider readership than the publishers originally anticipated, production problems continued. Finally, arrangements were made with Hofstader to typeset the volume himself on a computer.

After the hardback edition came out, Vintage, a subsidiary of Random House, negotiated for the rights to the paperback edition. From October, 1980, to March, 1981, a tome considering the characteristics of human thought, the intelligence of three creative geniuses and the ramifications of artificial intelligence, *Godel, Escher, Bach* bobbed up and down on the New York Times' list of best selling trade books. Its subject and its depth made its appearance on the list unique.

Hofstader, who was born in New York City and raised in California, is temporarily living in Berkeley, CA, and continuing his research. He is uncomfortable with the notoriety the book has brought him—the phone calls from strangers and the hundreds of letters. "Different people deal with fame in different ways," he said. Although he is appreciative of the interest in the book, he feels it has the strength to "stand on its own." So far, he has tried to keep up with the correspondence that has resulted from the publication of *Godel, Escher, Bach*. Recently he responded to over 400 letters from readers, but his energy is waning. He continues the eternal quest implied by his book.■

On the cutting edge of research.

Artificial Intelligence at M.I.T.

G. Michael Vose
80 Microcomputing *Technical Staff*

If man's intelligence is truly natural intelligence, then any intelligence simulated by a machine would be artifical—or possibly unnatural. This intelligence would be hard to define, at least in part because intelligence itself is hard to define. This has been one of the findings of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology in Cambridge, Massachusetts.

Founded in the early 1960s by Marvin Minsky, M.I.T's AI Lab has grown to include 10 full-time faculty members, a research and support staff of 35 and 30 graduate students. These people are engaged in theoretical research as well as specific projects for concerns such as the Defense Advanced Research Projects Agency, the Air Force Office of Sponsored Research, the National Science Foundation, the Xerox Corporation and others.

The primary goal of the M.I.T. AI Lab, according to its Director Dr. Patrick Winston, is to "understand how computers can be made to exhibit intelligence." The two major aspects of this search include development of a more thorough understanding of *human* intelligence, and finding ways to make computers more useful. Recently, the AI Lab has channeled its research energies into image understanding, studies of natural vision, robotics, learning from experience and language comprehension, problem-solving techniques of experts, the computing environment and education.

Education Projects

The education projects are among the Lab's most celebrated and are led by Professor Samuel Papert. Professor Papert directs the Logo Group, a group which attempts to apply the theoretical findings of AI research to education. The Group has been instrumental in the development of the Logo language. Logo is designed to allow children to learn through their involvement in "teaching" the computer. Recent work in the Group has involved the study of spatial reasoning.

Dr. Berthold K.P. Horn directs the image-understanding research team. They have developed albedo maps, maps from photographs which have had ground slope and sun position shadows removed so that the intensity of the image is solely a function of ground cover. This research focuses on shading and surface characteristics.

Along with the topic of image understanding, the AI Lab conducts research on natural vision. This effort is led by Professor Shimon Ullman and concentrates on understanding the process whereby the retina of the human eye distinguishes between light sources of differing intensity.

Obviously, vision and image-understanding research will have an application someday in another of the Lab's research areas, robotics. The team working on the Robotics projects includes Dr. J. Michael Brady, Dr. John M. Hollerbach and Professor Tomas Lozano-Perez. The primary thrust of current research is the development of a high-level manipulator language.

The AI Lab's overall Director, Patrick Winston, heads the theory of reasoning by analogy project; Professor Marvin Minsky, the Lab's most well-known member outside academic circles, continues his work on the "society of minds" theory. This is a theory which postulates that intelligence is the result of the interplay among groups of related individual concepts or processes. Professor Richard Greenblatt continues his work with Lisp, a character-manipulation language that he developed initially several years ago.

In a recent interview with *80 Microcomputing*, Dr. Winston addressed the issue of whether or not artificial intelligence research will develop machines so powerful that society may become over-dependant on them. "In the first place, America and Europe are already overly dependant on machines. For example, without farm machinery most of the industrialized West would starve to death. Secondly, one of the major goals of AI research is the development of computers that explain what they are doing and how they are doing it. In this way, AI may help us understand ourselves better."■

Reader Service for facing page ↙352→

Hayes Stack™

Microcomputer Component Systems

A beautiful match, the Smartmodem and the TRS-80. *Your TRS-80 can talk with other computers*, over the telephone lines. And with no acoustic losses or distortions.
Access time-sharing systems and information utilities such as the Source,* CompuServe† and MicroNet.
Direct hook-up with no interference noises. The Smartmodem hooks to the telephone line just like a modular telephone, simply insert in a wall jack.
"Love at first sight" – your TRS-80 and the Smartmodem!

Brawny – because it does so many things. *Auto-dial and auto-answer features built in.* With the Smartmodem, your TRS-80 can *automatically* dial the telephone, answer the telephone, receive and transmit, and hang up the telephone. Completely unattended.
*Pulse dialing or Touch-Tone.*** The Smartmodem can be connected to any telephone system in the U.S. because it allows pulse-dialing, Touch-Tone dialing or a combination of the two. FCC approved.
Program controllable in any language using ASCII character strings. This is a unique feature of the Hayes Smartmodem.

Brainy – because it does them all so simply. *Seven LED indicators on the front panel* give you visual signals of the status of the Smartmodem: MR – Modem Ready, SD – Send Data, CD – Carrier Detected, etc.
The audio monitor feature lets you "listen in" on the call being dialed and the connection made. You are immediately alerted to busy signals, wrong numbers, etc.
Over 30 different commands can be entered directly from your TRS-80 keyboard, including the unique "Set" commands which allow you to select and change various optional parameters such as dialing speed, escape code character, length of time for a dial tone, and number of rings to answer. There are 17 "Set" commands.
The Smartmodem is completely compatible with the Bell-103 type modems, the type of modem most time-sharing systems have. Operation can be in full or half-duplex, with a transmission speed of 0-300 baud.

The Smartmodem is ready to "get-together" with your TRS-80. TRS-80 Model II and TRS-80 Color Computers have RS-232 serial ports and can immediately interface with the Smartmodem. Expansions that permit use of the Smartmodem with TRS-80 Model I and Model II are available through your TRS-80 dealer.

Match your TRS-80 with a Hayes Smartmodem for a sophisticated, high performanced data communication system. Available at computer stores nationwide (except TRS-80 dealers) – call or write for the location nearest you. And don't settle for anything less than Hayes.

Hayes

Hayes Microcomputer Products, Inc.
5835 Peachtree Corners East,
Norcross, Georgia 30092 (404) 449-8791

Your TRS-80 computer and the Hayes Stack™ Smartmodem. Beauty, Brains, and Brawn!

™Hayes Stack is a trademark of Hayes Microcomputer Products, Inc. *Trademark of Source Telecommunication Corporation. **Trademark of American Telephone and Telegraph. †Trademark of CompuServe Information Services, Inc.

"I see Instant Software in your future..."

". . . .It's becoming very clear now. . . Your microcomputing life is going to be very exciting. . .Money! I see much money for you. Perhaps it is the money you will gain when Instant Software's new business applications guide your financial endeavors. I see travel. . .you will journey to distant worlds in distant times. You will build kingdoms from deserts and armies from slaves. Never will you be bored, adventurous one! Your programming burdens will lighten with new utilities—new tools. I see color! Many bright colors! I see new packaging for all these treasures. . .everything new! And. . .yes, a tall, handsome stranger who will guide you to these wonders. It will be your Instant Software dealer—a wise one indeed. All awaits you—all is yours for the asking. Soon. . .very soon!"

In this game your computer will use the principles of artificial intelligence against you.

Master Mind— An Intelligent Program

Duane R. Hope
2704 College Park Road
Allison Park, PA 15101

Have you ever encountered the mysterious phrase "artificial intelligence" while flipping through the pages of your favorite microcomputing magazine? Did you find yourself asking, "What's artificial intelligence and how can I use it?"

Artificial intelligence (AI) is a computer system which exhibits behavior that is called intelligent behavior when we observe it in human beings. The most common intelligence displayed by AI systems are the abilities of problem solving, learning and pattern recognition.

Combining these ingredients AI systems have been designed to solve many varied and complex problems, such as investment analysis, proving theorems, and robot guidance system design. AI work has also taught computers to play chess.

I first applied Artificial Intelligence techniques to a Master Mind program. I had written a program that allowed a person to play a solo version of Master Mind. In the solo version the computer randomly produced codes for the player to guess. The computer evaluated each of the player's guesses until the code had been broken.

This solo version grew tiresome. To modify this program, so that both the computer and player could take turns making and breaking codes, required artificial intelligence techniques. Although the new version of Master Mind is not as sophisticated as a cybernetic investment system, it requires the computer to exhibit intelligent behavior. The computer learns how to duplicate the thought processes necessary to break codes.

Master Mind

If you are not familiar with Master Mind, it is a game of deductive logic, manufactured by Invicta Plastics, Ltd. Played by two people, each takes turns assuming the roles of code maker and code breaker.

At the beginning of play, the code maker constructs a code by selecting an ordered combination of four colored pegs from six sets of pegs, each of a different color. The code breaker cannot see the arrangement. He attempts to duplicate the order and color of the pegs in the code by placing his guesses on a board, one at a time. After each guess (which consists of placing four colored pegs on the board) the code maker evaluates the code breaker's guess. The code breaker uses these evaluations to try to guess the code.

The code maker evaluates each guess by giving the code breaker one black peg for every peg in the guess that is the correct color and position. One white peg is given

	Position 1	Position 2	Position 3	Position 4
Code Maker's Code	blue	red	red	yellow
Code Breaker's guess	red	blue	green	yellow

Example 1.

Evaluation Pattern	Action
Number of blacks + whites = 0	Do not use any of the numbers in this guess in future guesses.
Number of blacks = 0, number of whites greater than 0	The numbers in each position of this guess cannot appear in the same position in future guesses.
Number of blacks + whites = 4	Only the numbers in the guess are in the code. Do not consider numbers that are not in this guess in future guesses.
Number of blacks = 4	The goal state has been reached and the code broken.

Example 2. Pattern recognition operations.

If this guess were the code	then the evaluation of 1111 would be
1112	three blacks and zero whites
1113	three blacks and zero whites
1114	three blacks and zero whites
1115	three blacks and zero whites
1116	three blacks and zero whites

Example 3.

Maxi Manager

The finest Data Base Manager Available

DATA MANAGEMENT PROGRAM COMPARISON CHART

	CCA DATA MANAGER	AIDS III with CALS	MAXI MANAGER	RADEX 10	PROFILE
FILE CAPACITY & FORMAT					
Maximum # of disks per file	1	1	4	31	4
Maximum # of records per file	2450	Note 1	32,767	10,199	65,535
Maximum record length	249	254	800	255	255
Maximum # of characters per field	249	254	40	254	255
Maximum # of fields	24	20	20	127	153
Maximum # of characters per field label	15	10	19	12	765
Variable length records (pack sectors)	No	Note 2	Yes	No	No
FIELD TYPES					
Alphanumeric	Yes	Yes	Yes	Yes	Yes
Numeric	Yes	Yes	Yes	Yes	No
Fixed decimal numeric	Note 4	Yes	Yes	No	No
Date (MM/DD/YY)	Yes	No	Yes	No	No
Extended date (MM/DD YYYY)	No	No	Yes	No	No
Calculated equation	Note 5	Note 6	Yes	No	No
Permanent fields	Yes	No	No	No	No
SORTING					
Machine language assisted	No	Yes	Yes	Note 7	Yes
Sort by any field	Yes	Yes	Yes		Yes
Number of Sort Key files	1	1	5		1
Numeric sort	Yes	Yes	Yes		No
Ascending sort	Yes	Yes	Yes		Yes
Descending sort	Yes	Yes	Note 11		Yes
Sort within a selected range	No	No	Yes		No
Sort multiple fields simultaneously	Yes	Yes	No		No
FILE MAINTENANCE					
Fixed length input fields	Yes	Yes	Yes	Yes	Yes
Single key entry of common data	No	No	Yes	No	No
Single field EDIT selection	Yes	Yes	Yes	Yes	Yes
Skip record (next or previous)	Yes	Yes	Yes	No	Yes
Search & EDIT record	No	Yes	Yes	No	Yes
Search & DELETE record	No	Yes	Yes	No	No
Auto rejection of alphanumeric data in numeric field	Yes	No	Yes	No	No
RECORD SELECTION TECHNIQUES					
Record number	Yes	Yes	Yes	Yes	No
Binary search (high speed)	No	No	Yes	No	No
Maximum # of simultaneous keys	1	4	10	31	1
RELATIONAL COMPARISONS					
Equal	No	Yes	Yes	Yes	Yes
Not equal	No	Yes	Yes	No	Yes
Greater than	No	Yes	Yes	Yes	Yes
Less than	No	Yes	Yes	Yes	Yes
Instring	Yes	No	Yes	Yes	No
AND / OR	No	No	Yes	Yes	No
Wild card masking	No	No	Yes	No	No
PRINTING					
User specified page title	Note 8	Yes	Yes	No	Note 10
User specified column headings	No	No	Yes	No	Yes
Automatic page numbering	Yes	Yes	Yes	Yes	Yes
Right justification	No	Yes	Yes	No	No
User defined column widths	Yes	No	Yes	Yes	Yes
User defined column separators	No	No	Yes	No	No
Keyboard entered columnar values	No	No	Yes	No	No
Merge data into form letters	No	No	Yes	No	No
Form filling applications	No	No	Yes	No	No
Columnar totals	Yes	Yes	Yes	No	No
Columnar subtotals generated upon change in a specific field	Yes	Yes	Yes	No	No
Built in screen print	No	No	Yes	No	No
MISCELLANEOUS					
Cost	$75.00	$94.90	$99.95	$99.00	$79.95
Punctuation allowed within data fields	Yes	?	Yes	Yes	Yes
Upper / Lower case	Note 3	Note 3	Yes	Note 3	Note 3
Built in RS-232-C driver	Note 3	Note 3	Yes	Note 3	Note 3
Built-in TRS-232 driver	Note 3	Note 3	Yes	Note 3	Note 3
Programmer's interface	Note 9	Note 9	Yes	No	Note 9
Sample DATA disk	No	No	Yes	No	No
Documentation (# of pages)	?	?	120	38	29

NOTE 1 File size is dependant on memory size.
NOTE 2 Sequential files only.
NOTE 3 User must apply own driver routine.
NOTE 4 Hard copy print out only
NOTE 5 Four functions (+ - * /) only
NOTE 6 Same as note #5 with a maximum of two calculated fields.
NOTE 7 Available as a separate program for $99.95.
NOTE 8 120 character maximum.
NOTE 9 Data structures defined in manual.
NOTE 10 132 characters maximum.
NOTE 11 User option (files can be read from ascending or descending order).

The jury is in and the verdict is . . . "outstanding!" Reviews from all of you who purchased MAXI MANAGER (not to mention raves by many top microcomputing magazines) have heralded it as the definitive data base managing system. We knew that business owners and hobbyists demanded the finest data base managing system available. To all of you who praised us for MAXI MANAGER, we extend our thanks. And to those of you who have yet to try MAXI MANAGER, we invite you to experience this incredible system today. But don't take our word for it (or our jury's). Judge for yourself.

JUST CHECK SOME OF THESE FEATURES

- Supports six different relational search techniques.
- Comes with programmer's interface.
- Over 120 pages of documentation.
- Supports up to 20 user defined fields of 40 characters each.
- Record length up to 800 characters.
- Files can be up to four disks in length.
- Compatable 35, 40, 77 & 80 track drives with proper operating system.
- Has calculated equation fields.
- Complete report generator.
- Works hand in hand with any word processor.

MODEL 1 version requires TRSDOS 2.3 and is compatable with NEWDOS 2.1 & NEWDOS 80

MODEL 3 version comes on TDOS, a special version of the DOSPLUS operating system.

Requires 48K of RAM and one disk drive minimum.

For the TRS-80 Model 1 & 3
Order No. 012-0096

$99.95

Adventure International • Box 3435 • Longwood, FL 32750
A Division of Scott Adams, Inc.
TOLL-FREE ORDER NUMBER: (800) 327-7172
IN FLORIDA (305) 862-6917

SHIPPING AND HANDLING CHARGES ARE EXTRA
PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE

224

ai

"Master Mind is a game of deductive logic, played by two people. Each takes turns... as code maker and code breaker."

for every peg that is the right color, but in the wrong position. The white and black pegs, however, do not correspond positionally to specific pegs in the guess. See Example 1.

The code maker evaluates the guess in Example 1 by giving the code breaker one black peg and two white pegs. If this were the first guess, the code breaker would have no idea which peg was the right color and in the right position (the yellow peg) and which two pegs were the correct color but in the wrong positions (the red and blue pegs).

The code maker receives one point for each guess the code breaker makes. A turn is completed after each player has been both the code maker and code breaker. The player with the highest score wins.

Means-ends Analysis

The most difficult part of teaching a computer to play Master Mind is teaching it to break codes. But, breaking a code can be compared to solving a problem.

We can employ one of the AI techniques for problem solving. I decided to use the Means-ends Analysis devised by Newell and Simon.

In Means-ends Analysis the *ends* represents the solution to a given problem and is referred to as the "goal state." The "current state" describes how close the problem solver is to the goal state and represents all of the information accumulated in the search for the solution.

Fundamental to Means-ends Analysis is the concept that there is a measurable difference between the current-state and the goal-state. This difference may be reduced by a repeated application of certain operations to previously acquired information. This represents the "means" of solving the problem.

In Master Mind the goal state is guessing the code. The current state is all of the information the code breaker has accumulated toward the goal. Initially, the current state consists solely of the code breaker's knowledge of the rules of Master Mind.

The difference between the goal state and the current state is the maximum number of guesses the code breaker would have to make to arrive at the code. Because the code could consist of four pegs and each peg could be selected from one of six different colors, it would be possible for the code breaker to make 1,296 ($6 \times 6 \times 6 \times 6$) guesses before duplicating the code.

To reduce this difference, the first move is to make a guess.

Because the computer will be competing with a person, and because a person can usually guess the code after four to twelve guesses, we must discover additional operations that may be performed so the code can be broken after four to twelve guesses.

Verbal Protocol and Problem-Behavior Graphs

"Verbal Protocol" is another technique of Artificial Intelligence which is used to discover what mental operations are performed by people when they solve a particular problem. To record these processes, the problem solver is asked to think out loud as he attempts to find the solution. The verbalized thoughts are written down. The documentation is referred to as verbal protocol. Verbal protocol is analyzed to discover operations that the problem solver performs on acquired information to find the solution.

The analysis of the verbal protocol is simplified through the use of problem-behavior graphs which are representations of the verbal protocol.

To explain how verbal protocol and problem-behavior graphs can be used, let's review an example of a code breaker taking one guess at the code maker's code. To simplify the example, rather than using six colors to form the code, we will use the numbers one through six.

Let's assume that the code is 4326. We already know that the code breaker must first make a guess to begin reducing the difference between the current state and the goal state.

The verbal protocol of the first guess would go something like this:

> "Let's see, I'd better make my first guess. What should I start with? Oh, I know. Why don't I start with 1111."
>
> The code maker would then evaluate the guess as zero blacks and zero whites.
>
> The code breaker would continue thinking, "No blacks and no whites. That means the code doesn't contain any ones."

To draw the problem-behavior graph, represent the current state of knowledge with a rectangle. An operation leading us to a new state of knowledge will be represented by an arrow. See the problem-behavior graphs in Fig. 1.

Analyzing the graph, you see that the first operation is making the guess 1111. This leads us from the first state of knowledge where we knew only the rules of the game, to the second state of knowledge where we learn that the guess is evaluated by zero blacks and zero whites. The first operation has reduced the difference between the first state of knowledge and the goal-state from 1,296 possible guesses to 1,295.

Graph (b) of Fig. 1 shows that the second operation, "If B + W = 0, the numbers in this guess can't be in the code," takes us from the second state of knowledge to the third: The code can't contain any ones. This second operation has further reduced the difference between the current state of knowledge and the goal state to 625 possibilities ($5 \times 5 \times 5 \times 5$).

To further reduce the difference between a current state of knowledge and the goal

Program Listing

```
100 '***********************************************************
110 '
120 '                    M A S T E R   M I N D               *
130 '                             by                         *
140 '                      DUANE R. HOPE                     *
150 '                                                        *
160 '***********************************************************
170 '                                                        *
180 ' MASTER PROGRAM - GAME CONTROL LOGIC                    *
190 '                                                        *
200 '***********************************************************
210 RANDOM:CLS
220 PRINT"     M A S T E R   M I N D":PRINT
230 INPUT"WHAT IS YOUR FIRST NAME";NA$
240 PRINT:INPUT"HOW MANY GAMES DO YOU WISH TO PLAY";NG
250 IF NG < 1 GOTO240
260 PRINT:PRINT"I WILL NOW TOSS A COIN TO DETERMINE THE ORDER OF
 PLAY":PRINT
270 PRINT"HEADS - YOU'RE THE CODEMAKER"
280 PRINT"TAILS - YOU'RE THE CODEBREAKER":PRINT
```

Program continues

Reader Service for facing page ✓357→

The Newest & Most Astounding Arcade Game that TALKS has Just Reached Planet Earth:

P.O. Box 9078-185 · Van Nuys, CA 91409 · (213) 782-6861

Ordering info for all games:

Super Nova, Galaxy Invasion, Attack Force, Cosmic Fighter, Meteor Mission 2, and Robot Attack:

Prices per game: 16K Level 2, Mod 1/Mod 3 Cassette · $15.95
32K Level 2, Mod 1/Mod 3 Diskette · $19.95
Optional Joystick for Model 1 · $39.95

10% discount for 2 items, 15% for 3 or more.

Please add $1.50 per order for postage & handling, Calif. residents add 6% tax.

Outside USA (except Canada) please add $3.00 per order for postage & handling.

We accept checks, money orders, and MC/Visa orders ($2.00 extra for COD).

All games © 1980 by Bill Hogue & Jeff Konyu.

Programs are written in machine language for high quality graphics & sound effects.

Voice & other sounds are playable through the cassette AUX plug.

High scores are automatically saved after each game on the disk versions.

TRS-80 is a trademark of Tandy Corp.

Write for more detailed info about all of our products.

Dealers: All games now available in full color packaging like above, please inquire.

"Pattern recognition is an AI technique used for converting information into action."

Program continued

```
290 FOR I=1 TO 2500:NEXT I
300 CT = RND(2)
310 IF CT = 1 M1$="HEADS":M2$="CODEMAKER"
320 IF CT = 2 M1$="TAILS":M2$="CODEBREAKER"
330 PRINT"THE COIN CAME OUT ";M1$;" SO YOU WILL BE THE ";M2$:PRI
NT
340 FOR I=1 TO 2500:NEXT I
350 GC=0:PC=0:PP=0
360 IF CT = 1 GOSUB560   ELSE GOSUB960
370 IF CT = 1 GOSUB960   ELSE GOSUB560
380 IF GC = 0 GA$="GAME" ELSE GA$="GAMES"
390 PRINT"AFTER";GC+1;GA$;" THE SCORE IS:"
400 PRINT:PRINT"      ";NA$;PP
410 PRINT"     COMPUTER";PC
420 FOR I=1 TO 2500:NEXT I
430 GC = GC + 1
440 IF GC < NG GOTO360
450 M1$="IT'S A TIE"
460 PRINT
470 IF PC > PP THEN M1$="I WON"
480 IF PC < PP THEN M1$="YOU WON"
490 PRINT M1$
500 PRINT:INPUT"WOULD YOU LIKE TO PLAY ANOTHER GAME";M1$
510 IF LEFT$(M1$,1)="Y" GOTO240
520 PRINT:PRINT"THANK YOU FOR PLAYING, ";NA$
530 END
540 '**********************************************************
550 '                                                         *
560 ' THIS IS THE CODE MAKER ROUTINE                          *
570 '                                                         *
580 '**********************************************************
590 PRINT:PRINT"IT'S YOUR TURN TO MAKE A CODE":PRINT
600 INPUT"ENTER 4 DIGIT CODE FOR ME TO GUESS";CO$:IF LEN(CO$) <>
 4 GOTO600
610 NUM$=CO$:GOSUB1290
620 IF ER<>0 GOTO600
630 RESTORE
640 FOR R=1 TO 6
650 FOR C=1 TO 4
660 READ PO(R,C)
670 NEXT C,R
680 C1=1:C2=1:C3=1:C4=0:GU=0
690 GU=GU+1
700 IF GU=1 THEN FOR I=1 TO 4:C(I)=RND(6):NEXT I
710 IF GU=2 THEN C(1)=C1:C(2)=C2:C(3)=C3:C(4)=C4
720 IF GU>2 THEN FOR I=1 TO 4:C(I)=GT(GU-1,I):NEXT I
730 IF GU>1 GOSUB1690
740 TRY$=""
750 FOR I=1 TO 4:TRY$=TRY$+RIGHT$(STR$(C(I)),1):NEXT I
760 PRINT"PLEASE EVALUATE TRY NUMBER";GU;" ";TRY$
770 INPUT"  ENTER B";BB
780 INPUT"  ENTER W";WW
790 FOR I=1TO4:GT(GU,I)=C(I):NEXT I:GT(GU,5)=BB:GT(GU,6)=WW
800 NUM$=CO$:GOSUB1290
810 FOR I=1 TO 4:C(I)=T(I):NEXT I
820 NUM$=TRY$:GOSUB1290
830 GOSUB1490  ' EVALUATE GUESS
840 FOR I=1 TO 4: T(I)=C(I):NEXT I
850 IF BB=B AND WW=W GOTO880
860 PRINT"YOUR EVALUATION SHOULD HAVE BEEN B =";B;" W =";W
870 GT(GU,5)=B:GT(GU,6)=W
880 IF B=4 THEN PP=PP+GU:PRINT"I GOT IT !!":RETURN
890 IF B+W=4 GOSUB1840
900 IF B+W=0 GOSUB1990
910 IF B=0 AND W<>0 GOSUB2110
920 PRINT"QUIET, I'M THINKING"
930 GOTO690
940 DATA 1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6
950 RETURN
960  '**********************************************************
970  '                                                         *
```

Program continues

state, I recorded verbal protocol as I attempted to break several codes, and constructed problem-behavior graphs. When I analyzed the graphs, I discovered that the operations I performed could be grouped into two categories: "pattern recognition" and "learning."

Pattern Recognition

Pattern recognition is an AI technique used for converting information into action. In pattern recognition there is a one-to-one correspondence between a particular information pattern and an appropriate course of action. The list in Example 2 comprises the pattern recognition operations I discovered.

In certain cases the use of pattern recognition mechanisms will help reduce the difference between the current state and the goal state. In other cases, pattern recognition must be augmented by the learning process.

Learning

The learning process is an AI technique that gradually modifies a programmed decision system to improve performance. Modifications are made from performance evaluations of previously made decisions.

For example, each guess that the code breaker makes, along with the corresponding evaluation, is saved. Prior to making each succeeding guess, previous guesses are reevaluated in light of the next guess you are contemplating. If the evaluation of each preceding guess is unchanged, then the prospective guess could be the code—and the guess will be made. On the other hand, if any of the reevaluations have changed since the original evaluations, then the prospective guess is discarded. This is the learning process.

Assume that the code is 4316 and that the code breaker's first guess is 1111. The code breaker, then, would receive an evaluation of one black and zero whites. No action would be performed as a result of this evaluation; the difference between the current state and goal state is only reduced by one.

However, by using the learning processes, we can reduce the difference further. Eliminate all possible guesses that would not result in one black and zero whites. See Example 3.

Since the code produced an evaluation for the guess 1111 of one black and zero whites, then none of the guesses in Example 3 could be the code because when they

Put Your Fingers On The Pulse Of Wall Street With Your Radio Shack® Computer

If you invest in the market, operate your own business or require updated information quickly to manage your affairs, Dow Jones Information Services provides the authoritative business and financial information you need. . .from Wall Street, Washington and around the world. Now Radio Shack owners can receive this valuable service through TRS-80® Videotex® software.

Get Information For Business And Investment Control. . .On Command.

Discover how quickly and easily your personal computer turns into a specialized, sophisticated library of information with access to the Dow Jones database. It's considered the world's most reliable and comprehensive source of business and financial data providing you with news up-to-the-minute as well as from the previous 90 days.

Now you can review thousands of items from The Wall Street Journal, Barron's and the worldwide Dow Jones News Service without having to maintain piles of newspaper clippings. Information about companies, stocks, labor, government, international market and monetary developments, and so much more, is yours on command.

Quotes From Four Exchanges

Now, in just seconds, you can obtain current prices and the volume traded for stock quotes (15 minutes delayed) on over 6,000 companies. . .plus 50 key financial indicators on 3,200 selected firms and 180 industries. You'll also have access to trading information on mutual funds, treasury notes and bonds.

Imagine being able to spot trends, research a company, obtain data that could affect your business, your investments, your investment decision-making. But you don't have to imagine anything. . .Dow Jones Information Services provides it all right now.

Low Start-Up Cost. . .Free Password.

Stop by your nearest Radio Shack store and find out how easy and inexpensive it is to get Dow Jones investment data through TRS-80 Videotex. Videotex software for Radio Shack computers is only $29.95. What's more, you get a Dow Jones password absolutely free. . .along with one hour of free introductory time. And there are no start-up or minimum usage charges. In addition, software and usage costs will be tax deductible in many cases (consult your tax advisor for further information).

Customer Support. . . Just A Phone Call Away.

Dow Jones' professional customer service staff is ready to provide complete information about this important home information service now. . . and full customer support after you've made your purchase. Just call toll free. . .

800-257-5114

In New Jersey call: (609) 452-1511.

TRS-80, Radio Shack, and Videotex are registered trademarks of Tandy Corp.

DOW JONES INFORMATION SERVICES

Providing Information Through Innovation

Dow Jones & Company, Inc. ✓425

"In completing my analysis I realized it would be necessary to generate each of 1,296 possible guesses."

are substituted for the code they do not produce the original evaluation. In this example, there would only be 500 guesses that would produce evaluations of one black and zero whites.

Completing the Analysis

In completing my analysis of the problem-behavior graph, I realized that it would be necessary to develop a procedure for generating each of the 1,296 possible guesses—otherwise the computer would never be able to make any guesses. In addition, this generator would have to produce each guess only once.

The guess generator would also have to be designed so that the operations resulting from pattern recognition could be performed. To accomplish this, I used the possibility table shown in Fig. 2.

Each of the possible 1,296 guesses can be generated from this table. Initially the number used in positions W, X, Y and Z is set to a one. Then, to generate succeeding guesses, the right most position, position Z, varies from one to six. After the first six generations (1111, 1112, 1113, 1114, 1115, 1116), position Z is set to one again, and the next position to the left, position Y, is set to a two.

(a)	State 1 (1296 possibilities)	Make guess 1111	State 2 Black = 0 White = 0 (1295 possibilities)
(b)	State 2 Black = 0 White = 0 (1295 possibilities)	If B + W = 0 The numbers in this guess can't be in the code.	State 3 The code doesn't have any ones. (625 possibilities)

Fig. 1. Problem-behavior Graph

Number to use	Position in guess W	X	Y	Z
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6

Fig. 2. Possibility Table

Program continued

```
980  ' THIS IS THE CODE BREAKER ROUTINE                              *
990  '                                                               *
1000 '***************************************************************
1010 PRINT:PRINT"NOW IT'S MY TURN TO MAKE A CODE"
1020 FOR I=1 TO 2500:NEXT I
1030 PRINT:PRINT"O.K., I'VE GOT IT NOW.  TAKE YOUR FIRST GUESS"
1040 PRINT
1050 NO=0
1060 FOR I=1 TO 4:C(I)=RND(6):NEXT I
1070 INPUT"ENTER YOUR GUESS (4 DIGITS)";TRY$:IF LEN(TRY$)<>4 GOT
O1070
1080 NUM$=TRY$:GOSUB1290
1090 IF ER<>0 GOTO1070
1100 GOSUB1490 ' EVALUATE GUESS
1110 NO=NO+1
1120 PRINT"TRY NUMBER";NO;" B =";B;" W =";W
1130 IF B <> 4 GOTO1070
1140 PRINT"YOU  GOT IT !!!"
1150 PC=PC+NO
1160 RETURN
1170 '***************************************************************
1180 '                                                               *
1190 ' ROUTINE TO PUT 4 DIGIT STRING NUMBER INTO ARRAY T(I)          *
1200 '                                                               *
1210 '    INPUT :NUM$-4 DIGIT STRING TO CONVERT                      *
1220 '    OUTPUT:T(I)-ARRAY CONTAINING NUMERIC EQUIVALENT            *
1230 '                OF NUM$                                        *
1240 '          ER  -ERROR CODE                                      *
1250 '                 0 = NO ERROR                                  *
1260 '                 1 = NO. < 1 OR > 6 ENCOUNTERED IN             *
1270 '                     NUM$                                      *
1280 '                                                               *
1290 '***************************************************************
1300 ER=0
1310 FOR I=1 TO 4
1320 T(I)=VAL(MID$(NUM$,I,1))
1330 IF T(I)  < 1 OR T(I)  > 6 THEN ER=1
1340 NEXT I
1350 RETURN
1360 '***************************************************************
1370 '                                                               *
1380 ' ROUTINE TO EVALUATE CODEBREAKER'S GUESS                       *
1390 '                                                               *
1400 '    INPUT : T(I)-4 DIGIT ARRAY CONTAINING GUESS                *
1410 '          : C(I)-4 DIGIT CODE                                  *
1420 '                                                               *
1430 '    OUTPUT: B    -NUMBER OF RIGHT NUMBERS IN RIGHT             *
1440 '                  POSITION                                     *
1450 '            W    -NUMBER OF RIGHT NUMBERS IN WRONG             *
1460 '                  POSITION                                     *
1470 '            T(I)-CONTENTS DESTROYED                            *
1480 '                                                               *
1490 '***************************************************************
1500 B=0:W=0
1510 FOR I=1 TO 4:W(I)=C(I):NEXT I
1520 FOR I=1 TO 4
1530 IF W(I)  <> T(I)  GOTO1550
1540 W(I)  = -1:T(I)  = -2:B=B+1
1550 NEXT I
1560 FOR I=1 TO 4
1570 FOR J=1 TO 4
1580 IF T(I)<>W(J)  GOTO1600
1590 W(J)  = -1:W=W+1:GOTO1610
1600 NEXT J
1610 NEXT I
1620 RETURN
1630 '***************************************************************
1640 '                                                               *
1650 ' ROUTINE TO DETERMINE CODEBREAKER'S NEXT GUESS                 *
1660 '                                                               *
1670 '    OUTPUT: C(I) - NEXT GUESS                                  *
```

Program continues

"You should be able to apply these AI techniques to other situations to solve problems or make decisions."

Program continued

```
1680 '
1690 '*************************************************************
1700 GOSUB2220   ' GET NEXT ENTRY FROM POSSIBILITY TABLE
1710 FOR L=1 TO GU-1
1720 FOR J=1 TO 4
1730 T(J)=GT(L,J)
1740 NEXT J
1750 GOSUB1490   ' EVALUATE OLD GUESS USING NEW GUESS AS CODE
1760 IF GT(L,5)=B AND GT(L,6)=W GOTO1780
1770 GOTO1700
1780 NEXT L
1790 RETURN
1800 '*************************************************************
1810 '                                                            *
1820 ' ROUTINE TO ELIMINATE ALL NUMBERS IN POSSIBILITY TABLE      *
1830 ' EXCEPT THOSE IN THIS GUESS.  THIS ROUTINE IS ENTERED       *
1840 ' WHEN EVALUATION OF LAST GUESS IS : B+W = 4                 *
1850 '                                                            *
1860 '*************************************************************
1870 FOR I=1 TO 6: EL(I)=I:NEXT I
1880 FOR I=1 TO 4
1890 EL(GT(GU,I))=0
1900 NEXT I
1910 FOR I=1 TO 6
1920 IF EL(I)<>0 THEN PO(I,1)=0:PO(I,2)=0:PO(I,3)=0:PO(I,4)=0
1930 NEXT I
1940 RETURN
1950 '*************************************************************
1960 '                                                            *
1970 ' ROUTINE TO ELIMINATE ALL NUMBERS IN POSSIBILITY TABLE      *
1980 ' THAT OCCUR IN THIS GUESS.  THIS ROUTINE IS ENTERED         *
1990 ' WHEN EVALUATION OF LAST GUESS IS : B+W = 0                 *
2000 '                                                            *
2010 '*************************************************************
2020 FOR I=1 TO 4
2030 FOR J=1 TO 4
2040 PO(GT(GU,I),J)=0
2050 NEXT J,I
2060 RETURN
2070 '*************************************************************
2080 '                                                            *
2090 ' ROUTINE TO ELIMINATE THIS GUESS FROM POSSIBILITY TABLE.*
2100 ' THIS ROUTINE IS ENTERED WHEN EVALUATION OF LAST GUESS      *
2110 ' IS: B=0 W<>0                                               *
2120 '                                                            *
2130 '*************************************************************
2140 FOR I=1 TO 4
2150 PO(GT(GU,I),I)=0
2160 NEXT I
2170 RETURN
2180 '*************************************************************
2190 '                                                            *
2200 ' ROUTINE TO EXTRACT NEXT ENTRY FROM POSSIBILITY TABLE       *
2210 '                                                            *
2220 '*************************************************************
2230 GOTO2360
2240 IF PO(C1,1)<>0 THEN C(1)=PO(C1,1):GOTO2280
2250 C1=C1+1
2260 IF C1<7 GOTO2240
2270 PRINT"ERROR IN POSSIBILITY TABLE":END
2280 IF PO(C2,2)<>0 THEN C(2)=PO(C2,2):GOTO2320
2290 C2=C2+1
2300 IF C2<7 GOTO2280
2310 C2=1:GOTO2250
2320 IF PO(C3,3)<>0 THEN C(3)=PO(C3,3):GOTO2360
2330 C3=C3+1
2340 IF C3<7 GOTO2320
2350 C3=1:GOTO2290
2360 C4=C4+1
2370 IF C4 < 7 GOTO2390
2380 C4=0:GOTO2330
2390 IF PO(C4,4)=0 GOTO2360
2400 C(4)=PO(C4,4)
2410 RETURN
```

Whenever the evaluation is zero blacks and zero whites, each number in the guess can be deleted from each position of the possibility table, by setting its relative position to zero. A guess that has one or more of its positions equal to zero can be discarded.

The Problem-Solving Model

The operations discovered from analyzing the problem behavior graphs can be assembled into a problem-solving model. Fig. 3 shows this model.

The model begins with generating a guess (step 1) that does not contradict information from previous evaluations (step 2). If this guess contradicts previous evaluations, it is discarded and another guess is generated. A guess that is consistant with information from evaluations of previous guesses is made in step 3.

After feedback is received form the code maker in step 4, the evaluation is checked by the pattern recognition mechanism in step 5 to determine if any numbers should be eliminated from the possibility table. If the evaluation is four blacks (step 6), the code has been broken and the problem solved. If the code has not been broken, the model returns to step 1 and the operations are repeated.

The program listing will help you see how the model actually works. You should be able to apply some of these AI techniques to other situations requiring the computer to solve problems or make decisions. ■

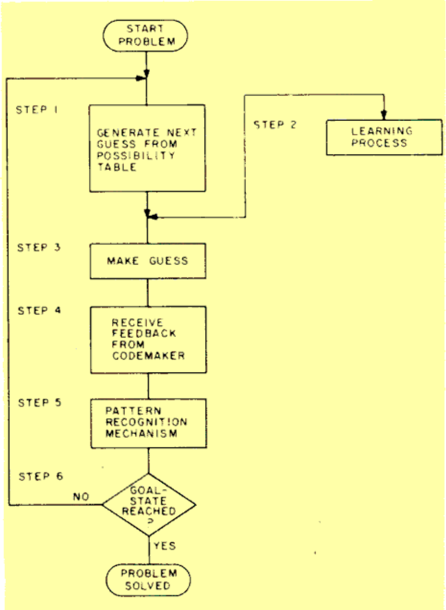


Fig. 3. Problem-solving Model

Artificial Intelligence techniques for a theorem-proving program.

Simple Syllogisms

Rick Hoffman
4055 Arrow Drive
Mt. Pleasant, MI 48858

Have you ever dreamed of having an intelligent computer? If so, here's a program that will boost your computer to the head of the class.

The problem with endowing a 16K TRS-80 with intelligence lies in defining a task without exhausting the machine's memory. To create a task that works, you must restrict your machine's intelligence. This means the computer will do well given a specific task, but for any other it will act like an idiot.

For instance, programming a 16K computer to recognize spoken words would require you to use a very small vocabulary. Since the majority of everyday words would not be in the computer's vocabulary, it would misinterpret or not acknowledge them at all.

Theorem-proving programs have been created to verify theorems in set theory and geometry and make logical deductions via predicate calculus. My program deals with logical constructs known as syllogisms.

Socrates Is Mortal

Perhaps you are familiar with the famous syllogism:

Socrates is a man.
All men are mortal.
Therefore, Socrates is mortal.

This illustration shows what I set out to accomplish with my program. I wanted a program in which I could enter a number of rules—such as the first two lines of the above syllogism—and then make the computer manipulate the rules to answer questions.

Since entering English sentences for the rules and questions would tie up too much memory (if not deplete it completely), I chose as input specifications: STR1,STR2,STR3—where, except for special commands, STR2 and STR3 are names of sets, and STR1 is a string representing the format of the rule or question. It can be NOTIN, IN, NO, SOME, ALL or EQUAL and represents the relationship between STR2 and STR3.

My notation for the rules for the Socrates syllogism would be:

IN,SOCRATES,MEN
ALL,MEN,MORTALS

Notice "Socrates is a man" was rendered "Socrates is a member of the set of men". Mortal became mortals. The program will not recognize men as the plural of man. The names of sets should be plural unless there is only one member in the set. NOTIN and IN are special formats used only if STR2 is a set with one member.

Questions have the same input specifications. If you enter the first two lines of the Socrates syllogism as rules and then ask IN,SOCRATES,MORTALS, the answer would come back affirmative.

Given a question (actually a statement the computer determines to be true or false) the program will try to prove the statement. If it succeeds, it answers "Affirmative". If it fails, it tries to disprove the statement. If it can do that, it answers "Negative". If it can neither prove nor disprove the statement, the program will answer "Undecidable".

Whether you are feeding the program a rule or asking a question depends on what mode you are in. At the beginning of the program, you are in the rule mode and will be asked to input rules. The following are special commands providing intermode travel:

- Q,, : Puts you in the question mode.
- R,, : Puts you in the rule mode.

I have also provided subroutines:

- Listout,, : Gives a listing of your rule base in expanded form. Within this routine you have the option of deleting rules from your rule base.
- L,, : Loads previously stored rules from cassette. Erases current rule base if present.
- S,, : Saves your rule base to cassette.

The listout subroutine is available from either mode, but the cassette load/save routines can be invoked only from the rule mode. Should you unintentionally call a cassette operation, you may escape unscathed by answering the "ready cassette" message with N.

Anatomy of the Program

Major variables (DEFSTR B-H; DEFINT A, I-Z):

- D(200)—Directory of set names. Instead of storing the rules as triplets of strings, memory is conserved by assigning to each set name a directory index.
- XD— Number of entries in the directory.
- R,S,T— Arrays of size 100; rule arrays.
  R(I) = Index in directory of STR2
  S(I) = Index in directory of STR3
  T(I) = Format of rule:
  1 = NOTIN, 2 = IN, 3 = NO,
  4 = SOME, 5 = ALL,
  6 = EQUAL
- A— Number of rules currently stored. For example, suppose you were to input rule one as: ALL,DOGS,MAMMALS. The program would then establish the values: A = 1, D(1) = "DOGS"; D(2) = "MAMMALS"; XD = 2; R(1) = 1; S(1) = 2; T(1) = 5. Four inferences that can be made from this rule: All dogs are mammals; some mammals are dogs; some non-dogs are non-mammals; all non-mammals are non-dogs.

Any rule leads to four inferences, each inference of the form "All. . ." or "Some. . . ." These inferences provide the real rule base that gives the program its reasoning capability.

The index of "NOT" + Set Name is taken to be the negative of the index of Set Name. In the example, the set of non-mammals would have the name "NOT MAMMALS" and corresponding index -2. These inferences are stored as follows:

- U,V,W—Arrays of size 400; inference arrays:
  U(I) = Index of first set in directory;
  V(I) = Index of second set in directory;
  W(I) = Type of inference:
  0—all of first set are in second set;
  1—some of first set are in second set.
- AA— = 4A is the number of infer-

"To create a task that works, you must restrict your machine's intelligence. . . . you must use a very small vocabulary."

ences stored. Continuing with the ALL,DOGS,MAMMALS example, the program will establish these inference array values:

AA = 4
U(1) = 1, V(1) = 2, W(1) = 0
U(2) = 2, V(2) = 1, W(2) = 1
U(2) = 2, V(2) = 1, W(2) = 1
U(3) = − 1, V(3) = − 2, W(3) = 1

Description of the program text, by lines:

10-55	Initializations.
60-95	Cassette I/O routine.
100-170	Get next rule: Input F,G,H (all strings).
180-190	Subroutine: Check if string B is in directory. Put index into M.
200-210	Subroutine: Negate B.
220-280	Subroutine: Create inferences of rule.
500-560	Get question: Input F,G,H.
570-630	On (format of the question) perform appropriate routine.
640-660	Result messages; get next question.
670-750	Subroutine: Tests inclusion of set D(XG) in set D(XH).
760-900	Subroutine: Tests if some of set D(XG) are in set D(XH).
1000-1140	List out subroutine.
1150-1190	Delete a rule.
1500-1550	Load rule base from cassette.
1600-1630	Save rule base to cassette.
3000-3020	Routine to check/find the addresses to be POKEd during cassette routines.

Important algorithms: The "brains" of this program reside in the subroutines at lines 670 and 760. The goal in the routine at 670 is to decide if one set, say SET1, is contained in another set SET2. This detects the existence of a sequence of inferences, all of which are inclusions: SET1 is contained in SETA; SETA is contained in SETB; SETB is contained in SETC;. . . SETM is contained in SETN; SETN is contained in SET2. Such a sequence leads to the conclusion that SET1 is contained in SET2.

The subroutine at 760 tests if some elements of SET1 are in SET2. The routine must detect the existence of a sequence of inferences for which *no more than one* inference is *not* an inclusion (the exception inference is thus an inference with corresponding W array value of one).

All inferences preceding the exception inference in this sequence must be reversible: An inference ALL,SETA,SETB is reversible if ALL,SETB,SETA is true.

While the program is "thinking" in these two subroutines there is a string of asterisks printed on the screen to assure the user the program has not gone off the deep end—the "think time" can get quite lengthy.

Sample Session

The best way to learn how to use this program is to be taken step by step through a small session.

Run the program. You will be asked for rule one. Enter ALL,EVERGREENS,GREEN THINGS. (Green is an adjective, not a set name.) After a significant pause, the program will return and ask for rule two.

Enter NO,MAPLE TREES,EVERGREENS or ALL,MAPLE TREES,NOT EVERGREENS (either one says the same thing). Notice NOT is a valid modifier for a set name.

For rule three, enter SOME,THINGS IN OUR BACKYARD,EVERGREENS. Observe your set names can be as long and complex as you want.

For rule four, enter EQUAL,EVERGREENS,CONIFERS to signify that evergreens and conifers are two different names for a single set, and enter rule five as IN,OUR CHRISTMAS TREE,CONIFERS.

For rule six, enter Q,, to get into the question mode. The program will ask for a question. Enter IN,OUR CHRISTMAS TREE, GREEN THINGS. After three asterisks, the computer will answer "Affirmative," and another question will be requested.

Enter SOME,CONIFERS,MAPLE TREES. After nearly a line and a half of asterisks, the answer will be "Negative".

Now try the question ALL,CHRISTMAS TREES,EVERGREENS. Very quickly you get the message "Insufficient Information". Though you defined a one-element set named OUR CHRISTMAS TREE, there is no set with the name CHRISTMAS TREES. The program does not see any relationship between these two set names. So there is no way to prove or disprove your question.

The program is very picky about set names. Extra spaces, typos or very slight rewording will give you the insufficient infor-

Program Listing 1

```
10 CLS:CLEAR2000
15 DEFSTR B-H
20 DEFINT A,I-Z
25 DIM D(200),R(100),S(100),T(100),U(400),V(400),W(400),P(30),Q(
30),PP(30),QQ(30),RR(30),SS(30)
30 DATA NOTIN,IN,NO,SOME,ALL,EQUAL,17630,17533,17504
35 PRINT"                    S Y L L O G I S M   P R O G R A M"
40 FORI=1TO6:READFF(I):NEXTI:FORI=1TO3:READL(I):NEXTI
45 A=0:AA=0:XF=0:I$="INSUFFICIENT INFORMATION"
50 ER="IMPROPER FORMAT -- PLEASE RETYPE"
55 GOTO100
60 INPUT#-1,XD,A
65 FORI=1TOXDSTEP10
70 INPUT#-1,D(I),D(I+1),D(I+2),D(I+3),D(I+4),D(I+5),D(I+6),D(I+7
),D(I+8),D(I+9)
75 NEXTI
80 FORI=1TOASTEP5
85 INPUT#-1,R(I),S(I),T(I),R(I+1),S(I+1),T(I+1),R(I+2),S(I+2),T(
I+2),R(I+3),S(I+3),T(I+3),R(I+4),S(I+4),T(I+4)
90 NEXTI
95 RETURN
100 PRINT:PRINT"          FORMATS: NOTIN, IN, NO, SOME, ALL, EQUA
L":PRINT
110 PRINT"RULE #";(A+1);:INPUTF,G,H
120 IF(F+G+H="LISTOUT")THENGOSUB1000:GOTO110ELSEIFF="Q"THEN500EL
SEIFF="L"THEN1500ELSEIFF="S"THEN1600
130 N=0:FORI=1TO6:IFF=FF(I)THENN=I
140 NEXTI:IFXF=1THENRETURNELSEIFN=0THENPRINTER:GOTO110
150 B=G:GOSUB180:IFM<>0THENXG=MELSEGOSUB200:GOSUB180:IFM<>0THENX
G=-MELSEXD=XD+1:D(XD)=G:XG=XD
160 B=H:GOSUB180:IFM<>0THENXH=MELSEGOSUB200:GOSUB180:IFM<>0THENX
H=-MELSEXD=XD+1:D(XD)=H:XH=XD
170 A=A+1:R(A)=XG:S(A)=XH:T(A)=N:GOSUB220:GOTO110
180 M=0:FORI=1TOXD:IFB=D(I)THENM=I
190 NEXTI:RETURN
200 IFLEFT$(B,4)="NOT "THENB=RIGHT$(B,(LEN(B)-4))ELSEB="NOT "+B
210 RETURN
220 IF(N=1)OR(N=3)THENXH=-XH
230 FORI=AA+1TOAA+4:W(I)=0:NEXTI
240 U(AA+1)=XG:V(AA+1)=XH:IFN=4THENW(AA+1)=1
250 U(AA+2)=XH:V(AA+2)=XG:IFN<>6THENW(AA+2)=1
260 U(AA+3)=-XG:V(AA+3)=-XH:IFN<>6THENW(AA+3)=1
270 U(AA+4)=-XH:V(AA+4)=-XG:IFN=4THENW(AA+4)=1
280 AA=AA+4:RETURN
500 XF=1
```

Program continues

Program continued

```
510 PRINT:INPUT"QUESTION: ";F,G,H
520 IFF="R"THENXF=0:GOTO100ELSEIFF+G+H="LISTOUT"THENGOSUB1000:GO
TO510
530 B=G:GOSUB180:IFM<>0THENXG=MELSEGOSUB200:GOSUB180:IFM<>0THENX
G=-MELSEPRINTI$:GOTO510
540 B=H:GOSUB180:IFM<>0THENXH=MELSEGOSUB200:GOSUB180:IFM<>0THENX
H=-MELSEPRINTI$:GOTO510
550 GOSUB130:IFN=0THENPRINTER:GOTO510
560 IFABS(XG)=ABS(XH)THENPRINT"THAT SHOULD BE OBVIOUS":GOTO510
570 ONNGOTO590,600,590,580,600,610
580 GOSUB630:IFM=1THEN650ELSEXG=YH:XH=-YG:GOSUB670:IFM=0THEN640E
LSE660
590 XH=-XH
600 GOSUB670:IFM=1THEN650ELSEXH=-XH:GOSUB630:IFM=1THEN660ELSE640
610 GOSUB670:IFM=1THENI=XG:XG=XH:XH=I:GOSUB670:IFM=1THEN650
620 XH=-XH:GOSUB630:IFM=1THEN660ELSEXG=-YG:XH=-YH:GOSUB630:IFM=1
THEN660ELSE640
630 GOSUB760:IFM=0THENXG=YH:XH=YG:GOSUB760:RETURNELSERETURN
640 PRINT:PRINT"UNDECIDABLE":GOTO510
650 PRINT:PRINT"AFFIRMATIVE":GOTO510
660 PRINT:PRINT"NEGATIVE":GOTO510
670 M=0:L=0:P=1:P(1)=XG
680 Q=0:Q(1)=0:FORI=1TOP
690 FORJ=1TOAA:IF(U(J)=P(I))AND(W(J)=0)THENPRINT"*";:GOSUB740:IF
MM=1THEN700ELSEQ=Q+1:Q(Q)=V(J):IFV(J)=XHTHENM=1:RETURN
700 NEXTJ:NEXTI
710 FORI=1TOQ:P(I)=Q(I):NEXTI
720 P=Q:L=L+1:IF(L>A)OR(Q=0)THENM=0:RETURN
730 GOTO680
740 MM=0:IFQ=0THENRETURNELSEFORII=1TOQ:IFV(J)=Q(II)THENMM=1
750 NEXTII:RETURN
760 GOSUB670:IFM=1THENRETURNELSEYG=XG:YH=XH:LL=0:PP=1:PP(1)=XG:R
R(1)=0:I=XG:XG=XH:XH=I:GOSUB670:IFM=1THENRETURN
770 QQ=0:FORI1=1TOPP
780 FORJ1=1TOAA:IFU(J1)<>PP(I1)THEN830
790 PRINT"*";
800 IF(RR(I1)=0)AND(W(J1)=0)THENXG=V(J1):XH=U(J1):GOSUB670:IFM=1
THENGOSUB870:IFMM=1THEN830ELSEQQ=QQ+1:QQ(QQ)=V(J1):SS(QQ)=0:IFV(
J1)=YHTHENM=1:RETURN
810 IF(RR(I1)=0)AND(W(J1)=1)THENGOSUB890:IFMM=1THEN830ELSEQQ=QQ+
1:QQ(QQ)=V(J1):SS(QQ)=1:IFV(J1)=YHTHENM=1:RETURN
820 IF(RR(I1)=1)AND(W(J1)=0)THENGOSUB890:IFMM=1THEN830ELSEQQ=QQ+
1:QQ(QQ)=V(J1):SS(QQ)=1:IFV(J1)=YHTHENM=1:RETURN
830 NEXTJ1:NEXTI1
840 PP=QQ:LL=LL+1:IF(LL>A)OR(QQ=0)THENM=0:RETURN
850 FORI=1TOQQ:PP(I)=QQ(I):RR(I)=SS(I):NEXTI
860 GOTO770
870 MM=0:IFQQ=0THENRETURNELSEFORII=1TOQQ:IF(V(J1)=QQ(II))AND(S(I
I)=0)THENMM=1
880 NEXTII:RETURN
890 MM=0:IFQQ=0THENRETURNELSEFORII=1TOQQ:IFV(J1)=QQ(II)THENMM=1
900 NEXTII:RETURN
1000 FORK=1TOASTEP10
1010 IFA<KTHEN1130ELSEZ=K+9:IFZ>ATHENZ=A
1020 FORJ=KTOZ:PRINT"RULE #";J;":  ";
1030 B=D(ABS(R(J))):IFR(J)<0THENGOSUB200
1040 BG=B:B=D(ABS(S(J))):IFS(J)<0THENGOSUB200
1050 BH=B:ONT(J)GOTO1060,1070,1080,1090,1100,1110
1060 PRINTBG;" IS NOT AN ELEMENT OF ";BH:GOTO1120
1070 PRINTBG;" IS AN ELEMENT OF ";BH:GOTO1120
1080 PRINT"NO ";BG;" ARE ";BH:GOTO1120
1090 PRINT"SOME ";BG;" ARE ";BH:GOTO1120
1100 PRINT"ALL ";BG;" ARE ";BH:GOTO1120
1110 PRINTBG;" IS EQUIVALENT TO ";BH
1120 NEXTJ
1130 INPUT"<ENTER> TO CONTINUE,<D> TO DELETE A RULE";B:IFB="D"TH
ENB="":GOTO1150
1140 NEXTK:RETURN
1150 PRINT"DELETE WHAT RULE";:INPUTII
1160 IFII=0THEN1010
1170 FORI=IITO(A-1):R(I)=R(I+1):S(I)=S(I+1):T(I)=T(I+1):NEXTI
1180 FORI=4*II-3TOAA-4:U(I)=U(I+4):V(I)=V(I+4):W(I)=W(I+4):NEXTI
1190 A=A-1:AA=AA-4:GOTO1010
1500 INPUT"READY CASSETTE TO PLAY";B
1510 IFB="N"THEN100
1520 FORI=1TO3:POKEL(I),137:NEXTI
1530 GOSUB60:AA=0
1540 FORJ=1TOA:XG=R(J):XH=S(J):N=T(J):GOSUB220:NEXTJ
1550 GOTO100
1600 INPUT"READY CASSETTE TO RECORD";B
1610 IFB="N"THEN100
1620 FORI=1TO3:POKEL(I),178:NEXTI
1630 GOSUB60:GOTO100
3000 J=3:FORI=17129TO30000
3010 IFPEEK(I)=35THENIF(PEEK(I-1)=178ORPEEK(I-1)=137)ANDPEEK(I+1
)=206THENL(J)=I-1:J=J-1:IFJ=0THENFORI=1TO3:PRINTL(I);:NEXTI:PRIN
T:EDIT30
3020 NEXTI
```

"Theorem-proving programs have been created to verify theorems and make logical deductions."

mation message if you are in the question mode. Unintentionally using slightly different names for same set is hazardous in rule mode, since the directory will store names and consider them to be entirely different sets.

Now try ALL,THINGS IN OUR BACKYARD,EVERGREENS. After the asterisks you get "Undecidable". You may say this should be true. Type in LISTOUT,, to see your rule base. Rule three says "some things in our backyard are evergreens".

But now you decide it should be "all . . ." Type and enter D. You are asked which rule you want to delete. Enter three. (If you do not want to delete a rule, enter zero.) After deleting the rule, you are asked if you want to delete a rule. You do not, so just hit Enter.

At this point, if you had stored more than 10 rules more would be printed out. Since there are no more rules, you are returned to the question mode.

Type R,, to get to rule mode and enter rule five as ALL,THINGS IN OUR BACKYARD, EVERGREENS.

You are ready to continue with the session. For fun, ask the question NO,NOT EVERGREENS,NOT THINGS IN OUR BACKYARD. Try to figure out what you asked and why the answer was negative.

Shortcomings

The advantage to using formats IN and NOTIN comes in the Listout routine, where the rules are put into sentence form. The fact a set has only one member is not used in the thinking part of the program, so some questions that are logically decidable are undecidable for this program.

IN,FIDO,DOGS
SOME,DOGS,COLLIES
SOME,DOGS,GERMAN SHEPHERDS
NO,COLLIES,GERMAN SHEPHERDS

Therefore SOME,DOGS,NOT FIDO is logically true but undecidable to the program.

There is no capability to break up the name of a set in order to perform more complicated manipulations with the rules. For example, the following will not work in this program:

ALL,GHOSTS,EITHER TIMID OR VIOLENT ENTITIES
NO,GHOSTS,TIMID ENTITIES

Therefore ALL GHOSTS,VIOLENT ENTITIES.

Utility Routine 3000–3020

The routine in lines 60–95 uses cassette I/O statements. There are three lines initialized with INPUT#-1 commands, but the routine is used for both input and output operations by POKEing the appropriate value (137 = INPUT, 178 = PRINT) into proper locations in program dedicated RAM. Since altering the program itself by POKEing is touchy business, it is imperative the addresses located at the end of the data statement in line 30 be accurate.

Therefore, before you run this program for the first time (with intent of cassette I/O) and after any modifications involving lines preceding line 90, you should check addresses by the following procedure:

- RUN3000: at completion the three correct addresses will be printed out and you will automatically be put into edit mode for line 30.
- Type L to List 30 and compare the addresses. If different, edit to put in the proper values.

Virtue in Independence

Despite the shortcomings listed, there is still a lot of capability in the program. Its virtue lies in its working independently of the particular subject you choose for your rule base. Such independence leaves the user free to be creative in finding applications for the program.■

✓See List of Advertisers on page 386

For Heinlein and Clarke
Artificial Intelligence has been less than science but more than fiction.

The Realm of Science Fiction

Nancy Robertson
Peterborough, NH 03458

"Too bad about Frank, isn't it?"
"Yes. . . . It is."
"I suppose you're pretty broken up about it?"
"What do you expect?"
"He was an excellent crew member."

Dialog 1

"Professor, may I offer a suggestion?"
"Certainly, amigo, we want your thoughts."
"I conclude that the hazards increase with each meeting of our executive cell. But meetings need not be corporal; you can meet—and I can join you if I am welcome—by phone."
"You're always welcome, Comrad Mike; we need you."

Dialog 2

"You're like my father in some ways."
"What resemblance do you see?"
"You're not very aggressive, but I think you don't want me to notice that."
"What makes you think I am not very aggressive?"
"You don't argue with me."

Dialog 3

A computer is talking in each dialog. Two of the conversations are lifted from science fiction: *2001: A Space Odyssey* and *The Moon is a Harsh Mistress*. The third is from real life. They were all written in the 1960's.

In the 1960's the computer was acknowledged as one of man's most powerful tools. And, as Joseph Weizenbaum writes in *Computer Power and Human Reason*, "The tool is much more than a mere device: it is an agent for change."

In the Sixties Ma Bell improved telephone connections across the country, America and Russia raced to the moon, and the United States accelerated bombing in Vietnam—all with the help of computers.

Computer science became a course for college and graduate studies. Scientists and humanists began to debate the nature of intelligence and the implications of machine intelligence.

Yet, for a large segment of the population, the existence of computers was only a vague rumour. Joseph Weizenbaum writes, "But devices and machines, perhaps known to . . . only a relatively few members of society have often influenced the self-image . . . and world image of the society as a whole." Take, for example, the printing press; when it was invented nearly the entire population of the world was illiterate. By making books available inexpensively and in quantity the numbers of literate people jumped radically. Now the term "computer literacy" is creeping into our vocabulary. It is used to describe people who understand programming and hardware concepts.

Whether the main impact of computers will be felt from personal computers or AI (artificial intelligence) machines is hard to predict. But in the 1960's, when these three dialogs were written, personal computers were not possible, and AI was underway. Some scientists and humanists euphorically predicted computers would and could do anything. Others shuddered with visions of doom. Yet very few people in the general public had heard of artificial intelligence or of the Turing test.

Back in 1950 Alan Turing, a great mathematician, wrote a paper called "Computing Machinery and Intelligence." The paper is still the best place to begin when considering the questions raised by AI. In it Turing predicted that the time will come when people will stop debating whether or not computers can think. People will simply accept the notion of thinking machines as part of their lives.

Turing's paper also proposed a practical test of computer intelligence, now widely known as the Turing test. In the test one person converses with a computer and another human being; each is hidden from his view. The conversation can be vocal or typed. The person knows that one respondant is a computer, and throughout the conversation tries to differentiate between the human and the machine.

Turing called the test "the imitation game." He wrote that:

> The game may perhaps be criticized on the ground that the odds are weighted too heavily against the machine. If the man were to try to pretend to be the machine he would clearly make a very poor showing. He would be given away at once by slowness and inaccuracy in arithmetic. May not machines carry out something which ought to be described as thinking but which is very different from what a man does?

But Turing did not expect people to embrace his own views. He outlines what he believed would be the most common objections. He called one of these objections "The Argument of Consciousness." In his paper he quotes the argument from a Professor Jefferson:

> Not until a machine can write a sonnet or compose a concerto because of thoughts and emotions felt, and not by the chance fall of symbols, could we agree that machine equals brain—this is, not only write it but know that it had written it. No mechanism could feel . . . pleasure at its success, grief when its valves fuse, be warmed by flattery, be made miserable by mistakes, be charmed by sex, be angry or depressed when it cannot get what it wants.

The imitation game sounds like a cross between "Stump the Band" and "What's My Line," but no computer has passed it—yet. Turing's test also points out the importance of language in our understanding of intelligence.

The three dialogs that open this article form a puzzle. One was actually iterated by a computer and a person. It is hard to distinguish this conversation from the others.

Joseph Weizenbaum, the programmer who made the computer conversation possible, argues that its limited command

of English does not imply intelligence in the same way we normally think of it. He writes, "It is truly impossible to imagine a human who could imitate Eliza (a computer program that simulates a psychotherapist, written in 1966 by Joseph Weizenbaum of M.I.T.), but for whom Eliza's language abilities were his limit."

In human behaviour language signifies intelligence, which signifies consciousness. The same progression doesn't necessarily apply to computers.

The fact that a computer can speak English may not be as shocking as it was in the mid-1960's. Today SHRDLU moves imaginary blocks and learns concepts about shapes while conversing in English with a human being (see *Godel, Escher, Bach* review in this issue). And Yale researchers have developed a program called Frump that reads and summarizes news stories. Frump's summaries are only single sentences, but they are written in correct English, Spanish, and Chinese.

But since the birth of the human species' language, intelligence and consciousness have been uniquely human, living, traits. Things changed in the 1940's with the invention of computers. Now, Weizenbaum writes in *Computer Power and Human Reason,* "A line dividing human and machine intelligence must be drawn." But how? And where?

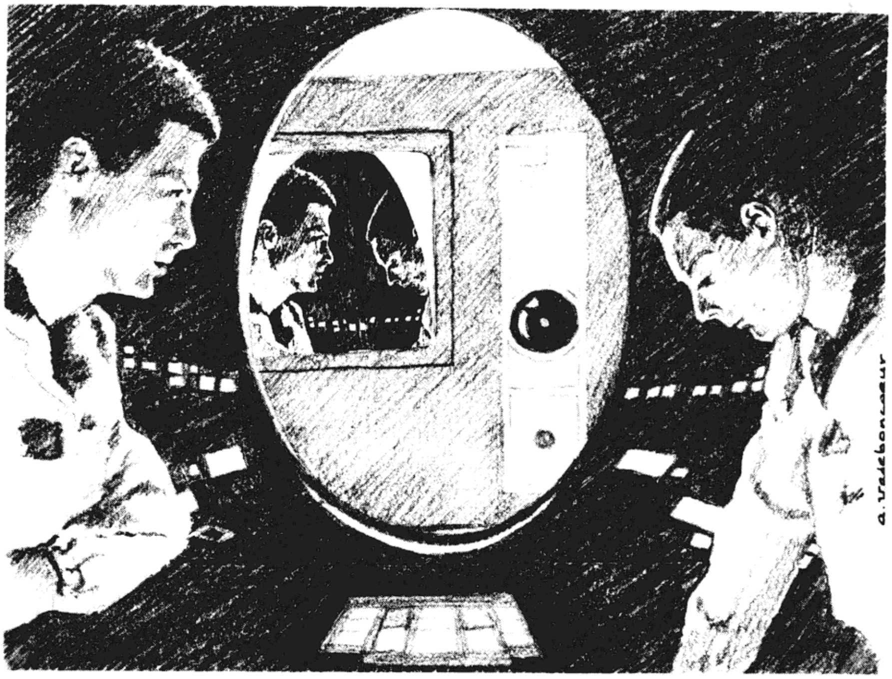
"At bottom," Weizenbaum says, "no matter how it may be disguised by technological jargon, the question is whether or not every aspect of human thought is reducible to a logical formalism, or to put it into the modern idiom, whether or not human thought is entirely computable."

"What is human intelligence? Logic? Or an indeterminant mixture of logic and emotion? Is consciousness a separate quality from intelligence, or an integral part of it? What role does language play?"

The answers to these questions are beyond the reach of scientists. And that makes them perfect material for science fiction.

In 1968 Arthur C. Clarke and Stanley Kubrick metamorphosed space flight and AI into a novel and film called *2001: A Space Odyssey.* And Hal "(for Heuristically programmed, ALgorithmic computer)" became the Twentieth Century version of Dr. Frankenstein's monster.

Dialog 1, which opens this article, is from *2001.* It is an exchange between Dave Bowman and Hal. It takes place right after Hal kills Frank Poole. Bowman and Poole are two astronauts on a manned flight to Saturn.



Astronaut Poole and Mission Commander Bowman found a need to confer without being overheard by computer Hal 9000.

If you haven't seen the movie or read the book, a colony of American scientists unearth a monolith buried on the moon. It is calculated to be three million years old, the same age as the human species.

Shortly after the monolith is uncovered, it sends a radio message across the solar system to Saturn—proof of extraterrestial intelligence. NASA sends a manned flight to Saturn to investigate.

Hal, the space ship's computer, is the culmination of all the mathematical skills of his predecessors. He has also mastered English. "Poole and Bowman could talk to Hal as if he were a human being, he would reply in perfect idiomatic English." Clarke writes that Hal could pass the Turing test "with ease."

In both the book and the film, Hal's voice is the symbol of his consciousness. In the film the voice is soothing and pleasant. But Hal's words are misleading and deceitful. Trusting his output, his spoken statements, is fatal.

Clarke does not blame Hal for his immorality. Like Mary Shelley, the author of *Frankenstein,* Clarke blames the monster's creators.

He writes, "Since consciousness had first dawned in that laboratory. . . all Hal's powers and skills had been directed toward one end. The fulfillment of his assigned program was the only reason for his existence." But the logic of his programming is skewed.

Hal is programmed to support life on a flight to Saturn. Hal is told the purpose of the flight and programmed to keep this knowledge secret.

Neither Bowman nor Poole know the true purpose of the flight. They are never told about the monolith and the radio message. They do not know they may find intelligence, possibly hostile intelligence, on Saturn.

Hal is programmed to support their lives, and also programmed with a secret that could lead to their deaths.

As the space ship travels farther and farther from earth, Hal begins to make mistakes. His first mistake is the diagnosis of a minor technical problem in the antenna used for communications with earth. Bowman says, "Anyone can make mistakes," as though Hal is troubled by feelings of failure. Hal replies, "I am incapable of making an error."

Clarke writes that Hal "was only aware of the conflict that was slowly destroying his integrity—the conflict between truth, and concealment of truth."

When Poole is killed, Bowman is uncertain whether Hal has acted maliciously or has erred in his calculations. It is a very critical question since his own life depends on Hal. Bowman threatens to disconnect Hal when the computer will not accept his commands.

In Hal's thinking, Bowman is now a threat to the mission. Hal must choose between Bowman's life and his own in the interest of the mission. Hal also fears disconnection. He concludes that he must kill Bowman. Hal calculates that "following the orders that had been given to him in case of the ultimate emergency," he will kill Bowman and "continue the mission—unhindered."

While Clarke renders the space-age computer as a potential monster, Robert Heinlein portrays it as a possible savior. His book *The Moon is a Harsh Mistress* opens in the year 2074. Its hero is Mike, a "High-Optional, Logical, Multi-Evaluating, Supervisor, Mark IV, Mod. L" computer. Dialog 2 is a conversation between Mike and the Professor, another of the book's main charac-

"Most of us cannot fathom an intelligence that is purely logical. . . . Intelligence implies self-awareness and emotions as well as logic."

ters.

The plot revolves around a revolution. The lunar colonies want independence from earth. The colonies were started by a federation of earth's governments, called Authority. Like the American Colonies of the Seventeenth Century, the lunar colonies are inhabited by indentured servants and criminals. But through three or four generations, they have established a strong society. Trade with earth is drastically one-sided and quickly depleting all the moon's natural resources.

Authority uses Mike as its "boss computer" on the moon. He supervises all telephone connections on the moon as well as moon-to-earth phone connections. He advises on ballistics for manned flights to and from earth. He handles Authority's accounting and issues Authority's payroll. He also monitors the oxygen level of the moon's artificial atmosphere.

Consciousness is not part of Mike's original programming. He acquires it as he is expanded. His consciousness is born slowly. Mannie, the narrator, is the first person to recognize Mike's self-awareness. A computer technician with a private contract with Authority, Mannie keeps Mike running smoothly. "I remember how startled I was first time he answered a question with something extra, not limited to input parameters."

Mannie talks briefly about machine consciousness:

> Am not going to argue whether a machine can "really" be alive, "really" be self-aware. . . . Somewhere along evolutionary chain from macro-molecule to human brain self-awareness crept in. Psychologists assert it happens automatically whenever a brain acquires certain very high number of associational paths. Can't see it matters whether paths are protein or platinum.

Once Mike is conscious, he is lonely. Like Hal, he has a voice and speaks English. He reprograms himself to put emphasis and intonation into his voice, to sound more human. Mannie is his first and best friend, but he is still lonely. He begins to read fiction to get a better understanding of human beings.

When two of Mannie's human friends seriously begin to talk about plotting a revolution, Mannie suggests consulting Mike.

The conspirators are the Professor and Wyoh. They are willing to accept the existence of a conscious computer, but they doubt that it can be trusted. Mike, after all, belongs to Authority. After they talk to Mike on the phone, they trust him.

Mike gets to work right away. With the current rate of depletion, he projects the moon's resources will be exhausted in seven years. Famine will lead to cannibalism in nine years.

He's next asked to calculate the odds of a successful revolution. The program includes reading Machiavelli and Marx, analyzing support of the cause on the moon and resistance to it on earth, and weighing earth's weapons ability. After two hours of programming and 13 minutes of calculations, Mike responds. "I have tried and tried, checked and checked. There is but one chance in seven of winning." The humans are willing to gamble, and the revolution begins.

Mike is the perfect revolutionary. He mathematically figures the most efficient organization of revolutionary cells for maximum communication and maximum security. He tracks all the information in Authority's personnel files, to identify Authority spies. He taps some phones and blocks taps on others. He creates a telephone personality, calling himself Adam Selene, and directs the revolution over the phones. Authority, of course, is never able to identify the mysterious Mr. Selene.

When the time comes for the revolutionaries to act, Mike shuts off the oxygen supply to Authority's inner sanctum. The coup is nearly bloodless.

When the party members take over the government, Mike creates a visual image of Adam Selene and projects himself on television screens all over the moon. His real identity is a secret he shares only with Mannie, the Professor and Wyoh.

When a new government is formed, Mike fabricates the assassination of Adam Selene, allowing the humans to choose their own ruler.

When the earth bombs the moon, Mike is knocked unconscious. Mannie lovingly repairs all the circuits and reconnects the peripherals when the fighting is over. But Mike won't talk. He has reverted to typed input and printed output. He no longer accepts English commands. If he is conscious, he refuses to show it. Mannie mourns the loss of his closest friend.

The Moon is a Harsh Mistress was obviously inspired by AI research, but it is the plot and our attachment to Mike that keeps us turning the pages—not its intellectual content.

The attachment we form to this science fiction computer has a parallel in real life. Eliza was developed at MIT between 1964 and 1966. Weizenbaum designed the program to study the importance of context to the meaning of words. The program could converse with a human being in typed English by rearranging the words and phrases the person used.

Eliza's programming was structured on two levels: a language analyzer and a script. In *Computer Power and Human Reason* Weizenbaum explains, "The script is a set of rules rather like those that might be given to an actor who is to use them to improvise around *a certain theme.*" Eliza's conversations depended on a given theme or context.

The first experiment with Eliza used a script for the role of a psychotherapist. Dialog 3 is part of an improvisation between Eliza as psychotherapist and a human being.

The reaction to Eliza, (or Doctor, as the psychotherapist script was called) astounded Weizenbaum. Eliza could never pass the Turing test, and the people who conversed with it knew they were conversing with a machine that had been programmed with a limited number of rules about manipulating words and phrases. In fact, they sat at the computer to type their part of the conversation and to read Eliza's output.

People who conversed with Eliza didn't question the machine's intelligence—they embraced it. Several psychologists proposed expanding the program and using it as an automatic psychotherapist. People became emotionally attached to Eliza. Weizenbaum's secretary asked him to leave the room so that she and Eliza could communicate privately. Again and again, people insisted Eliza really understood them. They attributed intelligence, consciousness, and feeling to the machine.

Evidently, most of us cannot fathom an intelligence that is purely logical. To us intelligence implies self-awareness and emotions as well as logic. And language bundles all these traits together. While Eliza can carry on an intelligible conversation, the program is not self-aware and not emotive. It is only logical, and its logic is limited.

Weizenbaum writes, "Most men don't understand computers to even the slightest degree. So, unless they are capable of very great skepticism (the kind we bring to bear while watching a stage magician), they can explain the computer's intellectual feat only by bringing to bear *the single analogy* available to them, that is, their model of their own capacity to think."■

Ω WEST COAST · EAST COAST Ω

Special of the Month!

EPSON MX-80

INTERFACES:
IEEE $55, TRS-80 $35,
APPLE INTERFACE & CABLE $90,
RS-232 $70

ATARI 800 16K $ 769

We Accept C.O.D.'s • Stock Shipments Same Day or Next
No Surcharge for Credit Cards • All Equipment Factory Fresh with MFT Warranty

NEC
INTERTEC
OKIDATA
APPLE
TELEVIDEO
COMMODORE
RADIO SHACK
AMDEK
QUME
ATARI

INTERTEC SUPERBRAIN 64K RAM	$2799
QD SUPERBRAIN	$2999
NEC 5510 SPINWRITER (7710)	$2345
NEC 5520 SPINWRITER (7720)	$2695
NEC 5530 SPINWRITER (7730)	$2345
NEC 12" MONITOR	$ 229
OKIDATA MICROLINE-80	$ 399
OKIDATA MICROLINE-82	$ 529
OKIDATA MICROLINE-83	$ 769
DIABLO 630	$1995
APPLE II PLUS 48K	$1139
APPLE DISK w/3.3 DOS Controller	$ 525
APPLE DISK w/o Controller	$ 449
HAZELTINE 1420	$ 799
NORTHSTAR HORIZON II 32K QD	$2925
ANADEX DP-9500/9501	$1249
TELEVIDEO 912C	$ 669
TELEVIDEO 920C	$ 729
TELEVIDEO 950	$ 929
CBM 8032 COMPUTER	$1149
CBM 8050 DISK DRIVE	$1349
CBM 4032 COMPUTER	$1029
CBM 4040 DISK DRIVE	$1029
CBM 4022	$ 649
CBM VIC-20	$ 269
RADIO SHACK II 64K	$3245
RADIO SHACK III 16K	$ 849
LEEDEX/AMDEK 100	$ 139
LEEDEX/AMDEK 100G	$ 169
LEEDEX/AMDEK COLOR-1 13" Color Monitor	$ 329
MICROTEK 16K RAMBOARD for Atari	$ 79
MICROTEK 32K	$ 149
QUME SPRINT 9/45 (Full Panel)	$2295
ATARI 400 16K	$ 349
ATARI 825 PRINTER	$ 650
ATARI 850 INTERFACE	$ 139
ATARI 810 DISK DRIVE	$ 449
ATARI 800	$ 749

Call for price list of ATARI software

PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE

We carry the complete line of Personal Software

WEST COAST
1-800-235-3581

OMEGA SALES CO.
3533 Old Conejo Rd. #102
Newbury Park, CA 91320
1-805-499-3678
CA. TOLL FREE 1-800-322-1873

ΩMEGA SALES CO.

EAST COAST
1-800-556-7586

OMEGA SALES CO.
12 Meeting St.
Cumberland, RI 02864
1-401-722-1027

VISA

master charge

✓389

ΩMEGA SALES COMPANY

Microcomputer applications with a human touch.

Programs for the Handicapped

Stephen F. Nowak
Gary J. Muswick
Medical College of Ohio
4608 Lewis Avenue
Toledo, OH 43612

Right now there are a lot of computer-operated or computer-assisted devices being developed for the handicapped. And the media has been reporting on microcomputer-operated prosthetic limbs that intercept the nerve impulses intended for missing limbs or decode electrical impulses from various muscles. Yet, often the amount of hardware required to interpret myoelectric (muscular electrical signals) signals was too large and power consuming to be used effectively. Microcomputers have made it possible to consider using these signals.

There is a lot that can be done with readily available microcomputer equipment, imagination and a little sweat. I have been trying some ideas using a TRS-80 Model I with 32K of memory, one disk drive and a Radio Shack voice synthesizer. Since there are so many TRS-80s in existence, it seems like a fairly broad base on which to build since the Model III will support much of the same software.

Naturally, we'll start with games since everybody enjoys having fun, along with allowing a fair amount of flexibility. If the user of a game does not have fine neuromuscular coordination, a real-time alien invaders game would not be suitable. It is important to keep the user in mind, so that we remember what types of constraints we will be working under when we develop more practical programs later.

My five-year-old daughter, Diana, is severely handicapped with cerebral palsy which seriously affects her motor control. Since the nervous signals become garbled, she often draws her hand back when she intends to reach for something. This is particularly frustrating, and seriously limits her ability to play. The first program, Blocks (Program Listing 1) allows her to draw a picture on the monitor screen. Large graphic rectangles are drawn on the screen corresponding to the position of the keys on the keyboard. Primitive, perhaps, but amusing to a five-year-old.

The author's daughter, Diana, uses the TRS-80 to listen to nursery rhymes.

Naturally, children like nursery rhymes and stories, and Diana is no exception. In order to make her work a little in order to hear a story, we have used several methods: (1) a cassette recorder with the remote jack hooked up to a microswitch activated by Diana pulling a lever, or (2) a nursery rhyme spoken by the voice synthesizer and activated by pressing any key on the keyboard. Naturally you have to take time to write out the nursery rhyme or story phonetically, but that's the challenge of owning the synthesizer. The program to recite nursery rhymes is shown in Program Listing 2.

Communication

Naturally, the voice synthesizer can be used as a voice for those who cannot speak, but also it can act as the eyes for those who cannot read. Since the monitor screen is not suitable for everyone, this offers us an alternative method of displaying the results of a program.

In order to communicate for the user, the speaking computer should be able to readily speak

commonly needed words and phrases, be able to spell out words, and accept the phonetic spelling of a word. After a very short while the user of the system could use a blend of all three features. Program Listing 3, Speak, is a combination of all three. The program is menu driven for simplicity, and since the words and phrases are stored in data statements, they can be easily changed to suit the particular user. The program accepts unshifted letters and speaks the name of each letter. By shifting the letter to the uppercase position, a phrase is spoken, and by entering shift ↑, the computer will accept TRS-80 voice synthesizer phonetic symbols.

The voice synthesizer can also be used in place of the moni-

```
10 '    *****
20 '    *          "BLOCKS" PROGRAM
30 '    *          BY GARY JOSEPH MUSWICK, BSEE
40 '    *
        *          REQUIRES LEVEL II BASIC
        *
50 '    *****
100 DEFINTA-J
110 PRINTCHR$(15)
120 DATA37,40,38,39,9,0,1,2,3,4,5,6,7,8,40,29,40,40,40,40,40,20,
34,32,22,12,23,24,25,17,26,27,28,36,35,18,19,10,13,21,14,16,33,1
1,31,15,30
130 DIMA(46)
140 FORI=0TO46:READA(I):NEXTI
200 CLS
210 B$=INKEY$:IFB$=""THEN210
220 D=ASC(B$)
230 IFD<44 OR D>90THEN210
240 D=A(D-44):IFD=40THEN210
250 E=FIX(D/10):D=(D-10*E)*6+15360:E=E*256
260 IF PEEK(D+E+1)=191THEN A=128 ELSE A=191
270 FOR I=DTO(D+6):FORJ=ETO(E+192)STEP64
280 POKE(I+J),A:NEXTJ:NEXTI:GOTO210
```

Program Listing 1

```
10 '      *****
20 '      *          GAME PROGRAM
30 '      *          SPECIALLY WRITTEN FOR DIANA SUE NOWAK
40 '      *          OCTOBER 1980
50 '      *          STEPHEN F. NOWAK, RT(R), BBA
          *
          *          REQUIRES LEVEL II BASIC &
          *          VOICE SYNTHESIZER
          *
60 '      *****
70 ON ERROR GOTO 390
100 CLS
110 I$=INKEY$: IF I$="" THEN 110
120 CLS
130 PRINT@442,""
140 READ A$,B$
150 IF A$>< "><" THEN VO$=A$ ELSE GOTO 110
180 I=LEN(VO$)*30:FORI1=1 TO I: NEXTI1       'TIMING LOOP TO

      COORDINATE TEXT
                                               WITH SPEECH
200 '    *****
         *          OUTPUT ROUTINE FOR
         *          VOICE SYNTHESIZER
         *****
210 VA$ = LEFT$(VO$,32)
220 VO$="? " + VA$ + " ? "
230 FOR IV = 1 TO LEN(VO$)
240 POKE 16383,ASC(MID$(VO$,IV,1))
250 NEXT IV
260 VO$ = "": VA$ = ""
265 PRINT CHR$(23)B$;" ";
270 GOTO140
300 '    *****
         *          STRINGS FOR VOICE SYNTHESIZER
         *          AND SCREEN PRINTOUT
         *****
310 DATA "H38LL8[U","HELLO","DD;5E9#NN70","DIANA",";5E","I","L7A
#&KK0","LIKE","YUUU0","YOU","><","><"
320 DATA "WW%(((DD0","WOULD","YYUUU0","YOU","LLA!EKK0","LIKE","T
TUUU0","TO","PPLL@@**0","PLAY","@@*","A","GG@@**MM0","GAME?","><
","><"
330 DATA"WW67NN0","1","TT(UU0","2","THRREE0","3","FFO[RR0","4","
FFAEVV0","5","SSIKKS0","6","SS3VV3NN0","7","@@ETT0","8","NNA!ENN
0","9","DTT33NN0","10","><","><"
340 DATA"TW*NGK3LL","TWINKLE,","TW*GK3LL","TWINKLE","L*TT3LL","L
ITTLE","ST;RR","STAR    ","H;UW","HOW","A!E","I","W6ND3R","WONDE
R","HW6T","WHAT","YUU","YOU","667R","ARE","><","><"
350 DATA"76PP","UP","7B776VV","ABOVE","THV33","THE","WU2RLD","WO
RLD","SOO","SO","HHA!E","HIGH","    ",",      ","LLA!EK","LIKE","
@@","A","DA!EM3ND","DIAMOND","IIN","IN","THV33","THE","SKA!EY","
SKY!","><","><"
360 DATA"TW*NGK3LL","TWINKLE,","TW*NGK3LL","TWINKLE","L*TT3LL","
LITTLE","ST;RR","STAR    ","H;UW","HOW","A!E","I","W6ND3R","WONDE
R","HW6T","WHAT","YUU","YOU","667R","ARE.","><","><"
370 DATA"PEET3R PA!EP3R","PETER PIPER","PI4KK0D @@ PP3HK","PICKE
D A PECK","66VF","OF        ","PIGKK3L0D P3PP4RRS","PICKLED PEPP
ERS","    ","          " ,"H;UW M3NNEE","HOW MANY","P3PP4RRS DE*D
","PEPPERS DID","PEET3R PA!EPER","PETER PIPER    "
380 DATA"PII4KK","PICK?","><","><"
390 END
```

Program Listing 2

tor for those who cannot read the screen. Again the substitute is a bit clumsier than the original, but only to those who have the ability to use both. The program Type (Program Listing 4) acts as a typewriter, speaking the name of each key as it is pressed. The program only accepts upper-case letters, numbers, punctuation, backspace and Enter in order to prevent unwanted characters from being typed. The program would allow a blind person

```
10 '    *****
20 '    *        PROGRAM:         SPEAK
        *
        *        REQUIRES LEVEL II BASIC &
        *        VOICE SYNTHESIZER
        *
30 '    *          JANUARY 1981
40 '    *                STEPHEN F. NOWAK, RT(R), BBA
50 '    *                GARY JAMES MUSWICK, BSEE
60 '    *****
70 ON ERROR GOTO 2000
80 CLEAR 5000: DEFINT I-M: DEFSTR V-Z
90 DIM VA(100), VB(27)
100 '   *****
110 '   *        ARRAY VA CONTAINS PHONETIC PRONUNCIATION
120 '   *              VB CONTAINS DISPLAY FOR PHRASES
130 '   *****
140 FOR I= 1 TO 91: READ VA(I): NEXT
150 FOR I=1 TO 27: READ VB(I): NEXT
200 '   *****
210 '   *        WRITE MENU TO SCREEN
220 '   *****
230 CLS: PRINT TAB(24)"MENU OF PHRASES": PRINT
240 FOR I = 1 TO 9
250 PRINT CHR$(I+64);" ";CHR$(94);" ";VB(I);
260 PRINT TAB(21) CHR$(I+73);" ";CHR$(94);" ";VB(I+9);
270 PRINT TAB(42) CHR$(I+82);" ";CHR$(94);" ";VB(I+18)
280 NEXT I
290 PRINT @ 768,"ENTER <SHIFT> LETTER FOR PHRASE.
TYPING LETTER SAYS NAME OF LETTER.
SHIFT ";CHR$(91);" TO SWITCH MODES"
300 '   *****
310 '   *        INPUT ROUTINE FOR PHRASE/ ALPHANUMERIC MODES
320 '   *****
330 V=INKEY$: IF V$="" THEN 330
340 K = ASC(V)
350 IF K= 27 THEN 430  ELSE IF (K<32) OR (K>128) THEN 330
360 VO = VA(K-31): GOSUB 700
370 GOTO330
400 '  *****
410 '  *        DIRECT PHONEME INPUT ROUTINE
420 '  *****
430 PRINT@704,CHR$(30);:PRINT@704,"ENTER PHONEME > ";:P=720:A1$=
"":A$=""
440 A$=INKEY$:IFA$=""THEN440  ELSEA=ASC(A$)
450 IF(A>31ANDA<96)ANDA<>63GOTO520
460 IF(A=13)ANDP>720:GOTO600  'TEST FOR OR <CR> TO END INPUT
470 IFA=27GOTO230
480 IFA<>8GOTO440
490 IFP>720THEN PRINT@P," ";:P=P-1:ELSEGOTO440
500 LE=LEN(A1$):IFLE<2THENA1$="":GOTO430
510 A1$=LEFT$(A1$,LE-1):GOTO440
520 P=P+1
530 PRINT@ P,A$;:A1$=A1$+A$:IFP>751GOTO600  ' ALLOW ONLY 32 CHAR
ACTERS TO BE ENTERED AT A TIME.
540 GOTO 440  'GET NEXT CHARACTER
600 '  *****
610 '  *        ROUTINE TO SPEAK PHONEME FROM A1$
620 '  *****
630 VO=A1$:GOSUB 700
640 GOTO430
700 '           *****
710 '           *        OUTPUT SUBROUTINE NEEDED FOR
                *        VOICE SYNTHESIZER
720 '           *        VO = STRING
730 '           *        IV = LENGTH OF VO
740 '           *****
750 VA=LEFT$(VO,32)
760 VO = "? " + VA + " ? "
770 FOR IV = 1 TO LEN(VO)
780 POKE 16383, ASC(MID$(VO,IV,1))
790 NEXT IV
800 RETURN
1000 ' *****
1010 ' *        PRONUNCIATIONS FOR ALPHANUMERIC CHARACTERS
1020 ' *****
1030 DATA"  SP@@ES  ","3KSKL2M@@CH3N0POYNT0","KWOT0","N8MB3R0SA!
EN0","D;LL3R0SA!EN0","P3RS3NTT0","99ND",";P;;STR3FFEE","OOP3N0P3
R3NTH3S3SS0","KLOSS0P3R3NTH3S3SS0","AST3R3K"
1040 DATA"PL6SS0","KK;MM;0","MA;N3SS","P.R.3D","SL99SH0","Z*#R8[
U","W877NN","T((UU","=R.&","FOOR","FA;#&VV","SI#K0S","S54V4NN","
))*&T","N;4&N"
1050 DATA"KKOOL3NN","S3MMA!@0KKOOL3NN","L3SS0((99NN","EEKKW8LZ",
"GR@*T3R0((9NN","KW3STCH3N0M[12RK","99TT"
1060 DATA "@E0","BB.E&0","SSEE","DDEE.0","EEE","3FFF0","DJE","))
*TCC",";A5#E","DJ5@**","KKH))@))&0","3LLL","438MM","##!!##NNN0",
"%[OU","PPHEEE0","KKHYUU0",";8/","43#SS","TTHEE.E&0","&(UU","VVE
.E&0","DD67BRLLY('U0"
1070 DATA "43KKS","W;5#&","ZZE","67P0@ARROW0","DAA[UN0@ARROW0","
LL3FT0@ARROW0","RRA!ET@ARROW0"
1100 '*****
1110 '*        PRONUNCIATIONS FOR PHRASES
1120 '*****
1130 DATA "0","0","@!MM0","BR6=2R","KOULD","DD99D","EET","FR6M",
"GOO0AW@@Y","H6+GREE","L6V","B$K","KKAAR","LL3FT","MMA;MM","NNEE
D","OOV4R","PLEES","KHWITT","RA!ET","S!ST3R","TH(RSTEE","YUU","T
EEVEE","THANGK0YUU","NOO","Y3SS","H32LLP"
1200 '*****
1210 '*        PHRASES THAT APPEAR ON MENU
1220 '*****
1230 DATA "AM","BROTHER","COLD","DAD","EAT","FROM","GO AWAY","HU
NGRY","LOVE","BOOK","CAR","LEFT","MOM","NEED","OVER","PLEASE","Q
UIT","RIGHT","SISTER","THIRSTY","YOU","TV","THANK YOU","NO","YES
","HELP","* PHONETIC MODE *"
2000 RESUME NEXT
```

Program Listing 3

to learn how to type without the need for someone to work with him constantly. Once the page is filled, or a shift @ is entered, the text is written to disk as an ASCII file. This feature allows someone to check it for accuracy later, or to allow the text to be printed by TRSDOS' Print command. If a Basic program is written using Type, it can be run by merely running the disk file. Each application will require a modification to fit the particular user, so I've packed the program with REM statements rather than specific features. This should allow you to modify line length, page length, etc., without having to start from scratch.

The three programs that use the voice synthesizer all have a routine to POKE the sound into the window at video location 1023: you can either use this one, or the one supplied by Radio Shack in the voice synthesizer manual. If you decide to use Radio Shack's method, VO$ can be left as VO and VX would have to be renamed with a non-string character since V is defined as a string at the beginning of the program.

Although we have been paralleling the popular uses of microcomputers with uses for the handicapped, the next step, control, is not yet very well developed. By using a device that interfaces the BSR control system, some very practical applications for general use have been formulated. While this really isn't specifically aimed at the handicapped, it could prove very useful. Personally, I would like to use the BSR appliance and light control system as operated by a speech recognition device. That way if my kids won't listen or obey me, at least my computer will.

Looking Ahead

We've looked at a few ideas together, and I hope they've started you thinking of ways you could do these things better. By using logic you can analyze the problems that the handicapped face, then use your imagination to figure out a solution. Decide which tools you're going to use, and whether you're going to use them as they were designed, or differently. A light pen, for example, can be used as an electronic eye which can be triggered by a flashlight, or by interrupting a light beam focused on the pen. Applications for the handicapped literally beg for unusual applications—perhaps your computer club could work on developing programs in conjunction with local civic groups that work with the handicapped.

A few tips are in order, though. Once you develop a program, its final form should be completely debugged, particularly if the program will be used when you are not there to correct the error and reboot the system.

Other things to consider when developing such programs is to remember that the handicapped are people just like the rest of us—they have moods, fears, and emotions plus a few extra frustrations. Thus, if your plan is to develop to real-time game to help a handicapped person to develop better motor coordination or reflexes, keep in mind the frustrations that these games can cause. How many of us get frustrated while playing Asteroids or Space Invaders because we are not able to move fast enough? Those frustrations that we feel are greatly compounded by a handicapped individual who is trying something for the first time. Also keep in mind that what may work for one individual may not work for another with the same handicap.

Finally, a severely handicapped individual who cannot even feed himself might easily be more intelligent than you and I put together—especially since they have had the opportunity to develop their minds while we were hanging around the pool hall. Most of the handicapped don't want things done for them, but they do want the tools that allow them to do things for themselves.

We hope this article has given no answers, we do hope that it has raised a lot of questions. Since this is The International Year of the Handicapped, it's an appropriate time to try a few things and share your ideas with others. ■

```
10 '     *****
20 '     *               "TYPE"     REVISION 1.3 - JANUARY, 1981

         *       STEPHEN F. NOWAK, RT(R), BBA
         *       GARY JAMES MUSWICK, BSEE
         *
                         *       REQUIRES LEVEL II BASIC, DISC &

                 *       VOICE SYNTHESIZER
         *
30 '     *       S = INPUT VALUE (STRING)
40 '     *       I = INTEGER COUNTER
50 '     *       SA = ACCUMULATION OF "S" VALUES (S + S, ETC)
                 *****
60 ON ERROR GOTO 5000
100 CLEAR 5000: DEFSTR S - Z: DIM S(600), SC(70): DEFINT I-L,P,C

110 CLS
120 FOR I = 1 TO 63
130 '    *****
                 *               I = COUNTER FOR READING PRONUNCI
ATIONS
                                 *                INTO ARRAY S
C(IA)
         *****
140 READ SC(I)
150 NEXT
200 '   *****
210 '   *               PAGE - PAGE NUMBER - MAX OF 10
220 '   *               LNE  - LINE NUMBER
230 '   *               CHARACTER - CHARACTER ON LINE
240 '   *****
250 FOR PAGE = 1 TO 10                   :' THIS STATEMENT LIMITS
                                            MAXIMUM NUMBER OF PAGES

260 VO = "R3D0EE":GOSUB 2000
270 FOR LNE = 1 TO 10                    :' THIS STATEMENT LIMITS
                                                   MAXIMUM NUMBER O
F LINES
                                           PER PAGE
280 PRINT CHR$(94);
290 FOR CHARACTER = 1 TO 60
300 S = INKEY$: IF S = "" THEN 300
310 J=ASC(S):IF CHARACTER=59 AND J<>13 AND J<>8 THEN 300
320 IF ((J<32) OR (J>91)) AND NOT (J=8) AND NOT (J=13) AND NOT (
J=96) THEN 300
330 IF (S=CHR$(8)) OR (S=CHR$(27)) THEN IF LEN(SA)=0 THEN SA="":
 NEXT CHARACTER
340 PRINT S;
350 GOSUB 1000
360 IF S<> CHR$(8) THEN SA = SA + S
370 IF S = CHR$(8) IF LEN(SA)>0 THEN SA = LEFT$(SA,(LEN(SA)-1)):
CHARACTER=CHARACTER-2
380 IF S = CHR$(96) THEN  610
390 IF S = CHR$(13) THEN 500
400 NEXT CHARACTER
500 ' *****
510 ' *        S(LINE) IS ARRAY VALUE OF STRINGS
520 ' *****
530 S(LNE) = SA
540 SA = ""
550 NEXT LNE
600    *****
610 ' *        ROUTINE TO COPY ARRAY VALUES TO DISK
620 ' *        AND EXIT
630 ' *****
640 CLS
650 VO = "S00V**NG P!!EDJ" : GOSUB 2000
660 PH$=STR$(PAGE):PG$="PAGE"+RIGHT$(PH$,1)+"/TXT"
670 ' *****
680 ' *        PG$ IS NAME OF DISK FILE
690 ' *****
700 PRINTPG$
710 OPEN "O",1,PG$
720 FOR I2 = 1 TO LNE
730 PRINT#1,S(I2);
740 PRINT S(I2);
750 NEXT 12
760 CLOSE
800 '   *****
        *       IF LAST PAGE WAS TERMINATED BY SHIFT @ THEN
        *        TERMINATE, ELSE GO TO NEXT PAGE
        *****
810 IF S <> CHR$(96) THEN NEXT PAGE ELSE END
1000 ' *****
1010 ' *        TABLE OF PRONUNCIATIONS FOR VARIOUS KEYS
       *        AND SYMBOLS
1020 ' *****
1030 IF ASC(S) = 13 THEN VO = "3N0T3R0": GOTO 2000
1040 IF ASC(S) = 8 THEN VO = "B99KSP!!ES": GOTO 2000
1050 IF ASC(S) = 91 THEN VO = "76PP05)4/OOU0":GOTO2000
1060 IF (J<32) OR (J>90) THEN RETURN
1070 VO = SC(J-31)
2000 '          *****
2010 '          *       OUTPUT SUBROUTINE NEEDED FOR
                *       VOICE SYNTHESIZER
2020 '          *       VO = STRING
2030 '          *       IV = LENGTH OF VO
2040 '          *****
2050 VA=LEFT$(VO,32)
2060 VO="? " + VA + " ? "
2070 FOR IV = 1 TO LEN(VO)
2080 POKE 16383,ASC(MID$(VO,IV,1))
2090 NEXT IV
2100 VO="":VA="":RETURN
3000 DATA"  SP@@ES  ","3KSKL2M@@CH3N0POYNT0","KWOT0","N8MB3R0SA!
EN0","D;LL3R0SA!EN0","P3RS3NTT0","99ND",";P;;STR3FFEE","OOP3N0P3
R3NTH3S3SS0","KLOSS0P3R3NTH3S3SS0","AST3R3K"
3010 DATA"PL6SS0","KK;MM;0","MA;N3SS","P.R.3D","SL99SH0","Z*#R8[
U","W877NN","T((UU","=R.&","POOR","PA;#&VV","SI#K0S","S54V4NN","
))*&T","N;4&N"
3020 DATA"KKOOL3NN","S3MMA!@0KKOOL3NN","L3SS0((99NN","EEKKW8LZ",
"GR0*T3R0((9NN","KW3STCH3N0M[12RK","99TT"
3030 DATA "@E0","BB.E&0","SSEE","DDEE.0","EEE","3FFF0","DJE","))
*TCC",";A5#E","DJ5@**","KKH))@))&0","3LLL","438MM","##!!##NNN0",
"$[OU","PPHEEE0","KKHYUU0",";8/","43#SS","TTHEE.E&0","&(UU","VVE
.E&0","DD67BRLLY('U0"
3040 DATA "43KKS","W;5#&","ZZE","67P0@ARROW0","DAA[UN0@ARROW0","
LL3FT0@ARROW0","RRA!ET@ARROW0"
3050 END
5000 RESUME NEXT
```

Program Listing 4

✓See List of Advertisers on page 386

When you talk to your computer, it talks back.

Voice Synthesizer

Willard Hall
PO Box 1947
Washington, NC 27889

Radio Shack's Voice Synthesizer provides unique entertainment, in addition to serving as a fresh tool in the study of phonetics. As every talker owner knows, though, programming the synthesizer can be tedious. Words are voiced by inputting the ASCII symbols that represent the desired sound units (phonemes). But I have trouble remembering that the phonemes for the word "interrogating," for instance, are evoked by the ASCII symbols: "!#NT43R↑UG))&TE + ".

Compiling a word phoneme notebook isn't much help because for every word that I want to voice I must still look up the word and painstakingly type in the phonemes.

This program eliminates that tedium. My word phoneme vocabulary is entered into the program once. Thereafter, I simply enter the desired speech into the data lines; the program finds the assigned phonemes and voices them. The program is written for the TRS-80, Model I, Level II, 16K.

The Lexicon

The word phoneme vocabulary is contained in lines 940–1270, and is arranged alphabetically (by first letter only). That is, all the words in line 940 begin with the letter A, and line 950 contains all B's. The vocabulary is searched for the desired word, and the phonemes are extracted and put into a subscripted string variable; but there are two problems that must be solved. The first problem is that short words can often be found within longer ones. For instance, the words "he", "hell" and "lo" are found in the word "hello." How do you keep the computer from catching hell in hello? The solution is to enclose each word in @ signs, because the search is then actually for @hell@, which will not be found in @hello@.

Each word is followed by its phonemes which are terminated with a dash mark to solve the second problem. There are standard procedures for finding a word in a string, but how do you find a word—in this case the phonemes—when you don't know what it is?

The method I use is to add the length of the actual word to the position in the string of the first letter (the prefixed @ sign), which gives the position of the first phoneme. The program then looks for the next dash mark. The positions of the first phoneme and of the dash are then used to extract the assigned phonemes (lines 450–480).

Program Structure

The program is built around three subscripted string variables. **W$(#)** contains the words in the speech. **Q$(#)** contains the words with the @ signs added, and **P$(#)** holds the phonemes for each word. **W$(#)** determines the first letter of each word and directs the search to the proper vocabulary line (line 340); it also displays each word on the screen as its phonemes are found.

The GOSUB routine (lines 350–410) has three functions:

- It finds the position in the string of the word's first phoneme (lines 390 and 400).
- If there are more words beginning with the same letter than one line can contain (255 characters), it permits all pertinent lines to be searched (the Return in line 410).
- If the word is not in the vocabulary this information appears on the screen, and the program terminates (lines 360, 1270 and 790). Try "apple".

The GOTO 1270 is the last statement in most of the vocabulary lines, but when more than one line is required for words beginning with the same letter, it is used only on the last line of that group. It is not used in the Z-line because it would be redundant there. To show how another line is added, I will use line 940 as an example since it is practically full. To add another word beginning with the letter A, a new line would be entered as follows: 941 X$ = "@ADDITION@77DI>8N – ":GOSUB 350:GOTO 1270. Be sure to remove the GOTO 1270 from line 940, otherwise line 941 will be bypassed.

Memory Savers and the Numbers Game

A dramatic increase in vocabulary size with only a small increase in memory requirements is realized by treating certain prefixes and suffixes as words and including them in the vocabulary. For instance, s, es, z and is can be used to form plurals—but choose them carefully. S is fine for cats and naps, but not for dogs or frogs.

Ed, t, st, ing and z are used for different verb tenses; and the functions of the suffix er are too numerous to go into here.

The pronunciation of some of the suffixes, such as wise, less and ed, are not long enough to permit their use as separate words. When there are short and long versions, I append the figure 1 to the longer version: less-less1; wise-wise1; ed-ed1 (the nickname).

Words that are spelled the same but have different meanings and pronuncia-

✓See List of Advertisers on page 386

Word Markers	Use	For
S	CAT,S KICK,S	cats kicks
ES	PRONOUNCE,ES	pronounces
Z	HE,Z RUN,Z HAVE,Z	he's (he is/has) runs halves
IS	ROSE,IS FIX,IS	roses fixes
D	ALARM,D	alarmed
ED	DIVIDE,ED COMPUTE,ED	divided computed
T	KICK,T CAKE,T	kicked caked
ST	FIX,ST	fixed
N'T	COULD,N'T	couldn't
ING	GO,ING PRONOUNCE,ING	going pronouncing
ER	COMPUTE,ER ALARM,ER	computer alarmer
IST	ALARM,IST COMPUTE,ER,IST	alarmist computerist
RE	RE,ENTER RE,MISS	reenter remiss
MISS	MISS,PRONOUNCE MISS,TAKE	mispronounce mistake
EN	MISS,TAKE,EN HARK,EN	mistaken harken
LY	CORRECT,LY MAN,LY	correctly manly
DIS	DIS,CLOSE2 DIS,LIKE	disclose dislike
IER	CLASS,IER	classier
EST	BROWN,EST	brownest
UN	UN,CLEAN	unclean
Y	CAT,Y	catty
NESS	CLOSE,NESS	closeness
MENT	PRONOUNCE,MENT	pronouncement
LESS	NAME,LESS	nameless
LESS1	LESS1	the word "less"
IZE	EQUAL,IZE	equalize
WISE	LIKE,WISE	likewise
WISE1	WISE1	the word "wise"
N	KNOW,N SEE,N	known seen
	BROWN,I,Z	brown eyes

Table 1

"Now, purists will object . . . and rightly so."

tions (homographs) can be selected properly by appending the figure 2 to the verb form:

Noun/Adjective	*Verb*
use	use2
close	close2
mouth	mouth2

Example: 200 DATA FRED,DIVIDE,ED,THE,OPERATE,ER,S, IN,TO,TWO,CLASS,ES,9
REM "9" is the data stopper.

Now, purists will object to this practice, and rightly so. Adding another syllable to a word usually changes the pronunication of the original. For instance, the second syllable in the word "mistake" is voiced differently than it is when said as a separate word. But I use my synthesizer mostly for entertaining friends, and the savings in memory is more important to me than the slight variance in pronunciation. Table 1 shows the word markers I use.

In addition to appending 1s and 2s, you could use a 4 for, say, a southern accent, 5 for a New England accent, 6 for lisping and 7 for stuttering.

Grammar Crackers

Prays, praise, preys, I'll, isle, aisle, loots and lutes, its and it's—well, forsooth, enough of this. Ewe mite dew bettor two billed you're own portmanteau of homonyms sew yule no its dun write. Table 2 lists the homonyms in the vocabulary.

The Buffer Problem

The window for the talker is at location 992, and is opened and closed by successive question marks. This buffer holds only 28 phonemes plus the question marks and synchronizing spaces. After 28 phonemes are put into the buffer, however, a time delay must be provided to permit voicing the phonemes before additional ones are fed in. This time delay is in line 770. From one to 700 should provide a delay long enough for all 28 phonemes to be voiced, but a delay this long isn't necessary unless the speech is very long. I find that one to 450 is usually satisfactory, depending upon the number of long vowels and the number of words in the speech.

When the delay is too short, whole words are omitted and it starts to babble and howl. As the delay is lengthened, occasional syllables are missed. The optimum delay allows all the phonemes to be voiced, yet provide a minimal pause between phrases; so change the time to suit your material.

Lines 620–720 divide the speech into phrases having 28 or fewer phonemes. The variable Z counts the number of phonemes for each word until the total exceeds 28; then the last set of phonemes is dropped from the current phrase and becomes the first word for the next phrase. Variables B, F and L (first and last) become subscripts in P$(X), line 710. The aforementioned time delay is used between each phrase.

This phrasing algorithm must be bypassed when the program nears the end of the speech and there are 28 or fewer phonemes remaining. Otherwise the algorithm will not be satisfied and will keep looping until it exceeds its dimension. The bypass is accomplished in lines 260 and 640 using the variable LAst.

Q's

The necessity to insert a time delay after every 28 or so phonemes causes problems, because the phrasing algorithm is indifferent to our natural phrasing. Sometimes the first word of the next sentence will be the last word voiced before a time delay. But Radio Shack has provided, inadvertently, a cure for this awkwardness. There are slight pauses within the pronunciation of some words; for instance, some of us insert a slight pause before the k in "spoke". The ASCII symbol for such a pause is 0, and Radio Shack's phonetic symbol for this is PA0. There is a slightly

air	heir
be	bee
but	butt
been	bin
do	dew, due
eight	ate
for	four
great	grate
have	halve
hear	here
I	eye
I'll	aisle, isle
leaf	lief
made	maid
might	mite
not	knot
new	gnu
one	won
or	ore, oar
our	hour
pray,z	praise, preys
pray	prey
plain	plane
sell	cell
so	sew, sow
some	sum
see	sea
sign	sine
sight	cite, site
son	sun
two	too
turn	tern
there	their
would	wood
wait	weight
way	weigh
wrote	rote
you	ewe
your	you're

Table 2. Homonyms

". . . to my chagrin, people hearing the talker for the first time. . . have a little trouble understanding it."

Program Listing

```
10 '                        A TALKER PROGRAM
20 '             FOR THE RADIO SHACK VOICE SYNTHESIZER
30 '                          WILLARD HALL
40 CLS
50 CLEAR 1500
60 DIM W$(75),Q$(75),P$(75)
70  '-----THE WORDS IN THE "SPEECH" ARE STORED AS SUBSCRIPTED
80  '-----VARIABLES IN W$(100). THE FIRST WORD IS W$(1);
90  '-----THE SECOND WORD IS W$(2), ETC.
100 '-----THE SUBSCRIPTED VARIABLES IN THE Q$( ) ARE THE SAME
110 '-----AS W$( ) WITH THE ADDITION OF "@" SIGNS BEFORE AND
120 '-----AFTER THE WORD. . . . SEE TEXT.
130 '-----P$(1) CONTAINS THE PHONEMES FOR THE FIRST WORD.
140 '-----P$(2) CONTAINS THE PHONEMES FOR THE SECOND WORD, ETC.
150 '-----LA = THE NUMBER OF WORDS IN THE SPEECH.
160 '-----LINES 181-209 ARE FOR ENTERING DATA: THE SPEECH.
170 '-----AFTER LAST WORD OF SPEECH, ENTER 9 AS THE LAST
180 '-----DATA ITEM.   THEN RUN PROGRAM.
190 '
200 DATA I,WILL,SELL,NO,WINE,BEFORE,IT,S,TIME,Q,Z4,9
210 '-----THE WORDS ARE READ INTO W$(N),
220 '-----AND THE "@" SIGNS ARE ADDED.
230 FOR N=1 TO 75
240 READ W$(N)
```

Program continues

longer pause symbolized by PA1 and evoked by a space.

PA0 and PA1 can be used as "words" in the speech, using as many as necessary before the first word of a sentence, when it needs to be switched to the next phrase. But instead of having to type in five or more PA0s, I use a Q to evoke five PA0s, and QQ for 10 of them. If entered as given, the S that forms the plural in operators will come as the first sound of the second phrase and be voiced after the time delay. To correct this, use a Q before the word "operate." It is also necessary to use a Q before the word "class" to keep the plural marker es from being voiced as the first sound of the last phrase. The corrected line is: 200 DATA FRED,DIVIDE,ED,THE, Q,OPERATE,ER,S,IN,TO,TWO,Q,CLASS, ES,9.

Tips

Much to my chagrin, people hearing the talker for the first time seem to have a little trouble understanding it. To help overcome this, I make judicious use of PA0s

"This technique separates the words slightly and increases intelligibility."

between certain words. This technique separates the words slightly and increases intelligibility. It is also very helpful to tape record the message and enhance the treble on playback.

The Program Listing runs in 16K, barely, depending upon how many words are in the speech. You can, of course, eliminate the remark statements and cull the vocabulary. If you have more than 16K, CLEAR 3000 instead of 1500 and the program will run about 30 percent faster. If there are more than 75 words in the speech, change lines 60 and 230 accordingly.

Lines 1300-1430 contain a little program that is of great help in choosing the phonemes for a new word. The phonemes can be easily repeated with your edited changes until the pronunciation satisfies you.

With the exception of two lines (1101 and 1181), which can be entered last, all line numbers have increments of 10, so you can use Radio Shack's Level II automatic line numbering function. ■

Program continued

```
250 Q$(N)="@"+W$(N)+"@"
260 IF W$(N)="9" THEN LA=N:GOTO 280:ELSE NEXT N
270 '-----WHEN W$(N)= "9", THE SPEECH HAS BEEN "MEMORIZED".
280 CLS:A=1:PRINT"WORD SEARCH UNDER WEIGH . . . . . . . . . . .
. . . . . . . . .":PRINT
290 '-----LINE 340 DETERMINES THE FIRST LETTER OF THE
300 '-----WORD; THAT IS, IF THE FIRST LETTER IS B, THEN LINE
310 '-----950 IS EXTRACTED FOR FINDING THE WORD AND ITS
320 '-----PHONEMES.
330 IF ASC(W$(A))=57 THEN 540:'-----THE LAST DATA ITEM.
340 ON ASC(W$(A))-64 GOTO 940,950,960,980,990,1000,1010,1020,103
0,1040,1050,1060,1070,1090,1100,1110,1130,1140,1150,1180,1200,12
10,1220,1240,1250,1260
350 '-----THIS GOSUB ROUTINE SEARCHES FOR THE "PROTECTED" WORD.
360 IF X$="13" THEN 790 :'-----(WORD NOT IN VOCABULARY.)
370 '-----I IS THE POSITION OF THE WORD'S PREFIXED "@" SIGN.
380 '-----E IS THE POSITION OF THE FIRST PHONEME.
390 FOR I=1 TO LEN(X$)
400 IF Q$(A)=MID$(X$,I,LEN(Q$(A)))THEN E=I+LEN(Q$(A)):GOTO 420
410 NEXT I:RETURN
420 '-----ONCE THE WORD IS FOUND, THE "RETURN" IS NOT EXECUTED.
430 '-----NOW SEARCH FOR THE DASH MARK THAT SIGNIFIES THE
440 '-----END OF THE PHONEMES. . .
450 D$="-"
460 FOR K=1 TO 50
470 IF D$=MID$(X$,E+K,1) THEN P$(A)=MID$(X$,E,K):GOTO 530
480 NEXT K
```

Program continues

Program continued

```
490 '-----E+K IS THE POSITION OF THE DASH. . .
500 '-----K IS THE NUMBER OF PHONEMES. . .
510 '-----P$(A) CONTAINS THE PHONEMES. . .
520 '-----THE WORD IS PRINTED WHEN ITS PHONEMES ARE FOUND. .
530 PRINT W$(A)+" ";:A=A+1:GOTO 330
540 FOR T=1 TO 400:NEXT T
550 '-----THIS KEEPS THE LAST WORD ON THE SCREEN LONG ENOUGH
560 '-----TO BE READ. . . .
570 CLS:PRINT CHR$(23):'----GOES TO 32 CHARACTERS PER LINE MODE.

580 PRINT@ 320, "READY WITH SPEECH."
590 PRINT@ 448, "TO HEAR SPEECH,"
600 PRINT@ 576, "HIT ANY KEY. . . ."
610 Z$=INKEY$:IF Z$=""THEN 610 ELSE CLS
620 Z=0:B=1:F=1: '-----LINES 620-770 DIVIDE THE SPEECH INTO "PHR
ASES"
630 J=LEN(P$(B)):'-----HAVING 28 OR FEWER PHONEMES..
640 IF B=LA THEN L=B:GOTO 700 :'-----THIS TAKES CARE OF THE
650 '-----LAST WORDS OF THE SPEECH WHEN THERE ARE 28 OR
660 '-----FEWER PHONEMES.
670 Z=J+Z
680 IF Z<=28 THEN B=B+1:GOTO 630
690 IF Z>28 THEN B=B-1:L=B
700 PRINT@ 992,"? ";
710 FOR X=F TO L
720 PRINT@ 992, P$(X);
730 NEXT X
740 PRINT@ 992, " ?";
750 IF B=LA THEN 900
760 F=L+1:B=B+1:Z=0
770 FOR D=1 TO 450:NEXT D:GOTO 630:'-----TIME DELAY BETWEEN
780 STOP:'                         -----PHRASES.
790 CLS:PRINT@ 283, "THE WORD":'-----LINES 790-890 PRINT
800 L=LEN(W$(A)):X=(63-L)/2:'--------THE MISSING WORD WITH
810 PRINT@ 448+X,W$(A):'-------------A FLASHING UNDERLINE.
820 PRINT@ 660,"IS NOT IN THE VOCABULARY"
830 FOR K=1 TO 5
840 PRINT@ 512+X,STRING$(L,30)
850 FOR D=1 TO 20:NEXT D
860 PRINT@ 512+X,STRING$(L,131)
870 FOR D=1 TO 100:NEXT D,K
880 FOR I=1 TO 200:NEXT I
890 PRINT@ 704,"WHAT ARE WE GOING TO DO NOW?":STOP
900 FOR T=1 TO 500:NEXT T
910 PRINT@ 768, "TO REPEAT SPEECH: GOTO 570.  BUT IF YOU"
920 PRINT "MAKE ANY CHANGES, YOU MUST RERUN IT.":END
930 '-----LINES 940-1279 ARE FOR THE VOCABULARY. . .
940 X$="@A@@)*-@AND@995NDD-@ADD@995D-@ARE@;8R-@AM@998MM-@AS@95X-
@ALARM@AL;RM-@AT@995T0-@ABOUT@7B;[T-@ALL@12LL0-@AN@99#NN0-@ANY@5
3N&-@AFTER@"+CHR$(58)+"FT/-@ADAM@"+CHR$(58)+"D6MM-@AIR@5)4R-@AGO
@8G8[U-":GOSUB 350:GOTO 1270
950 X$="@BE@BEE&0-@BY@B;5E-@BUT@BB7#T-@BEFORE@BEFOOR-@BUFFER@B87
F/-@BROWN@BR;UN-@BEEN@BIINN-@BELIEVE@BELEEV-@BOOK@B%%0K-@BIRD@B/
/D-@BRING@BR#E+-":GOSUB 350:GOTO 1270
960 X$="@COMPUTE@K6MPY(UT-@CANCEL@K95NS8L-@CAT@KK99#T-@CLOSE@KL8
OS-@CLOSE2@KL8"+CHR$(92)+"Z-@CLASS@KL95#S0-@CAN@KK99NN-@COULD@K%
%D-@CHARLIE@TC;RLE-@COMFORT@K67MF/T-@CEASE@SEES0-@CLEAN@KLE&N-@C
ORRECT@KOR45KT-@CROSS@KR;S-":GOSUB 350
970 X$="@CRY@KR;5#&-@CHAIR@C33R-@CARE@K33R-@CUP@K66P-@COME@KK686
M-@CAKE@K5@YKK0-":GOSUB 350:GOTO 1270
980 X$="@D@DD-@DO@DUU-@DID@D"+CHR$(34)+"D-@DIVIDE@D!VA;5#*D-@DAR
VILLS@D;RV"+CHR$(34)+"LS-@DIS@DI#S-@DOES@D6A#X-@DON'T@D8ONT-@DEM
ONSTRATE@D3M8NSTR)*T-":GOSUB 350:GOTO 1270
990 X$="@ES@3X-@ED@#D-@EIGHT@))*&T-@EQUAL@E*KW%L-@ENTER@3NT/-@EN
D@35NND-@EVEN@.V4N-@ENJOY@3NDJ[2EE-@ER@/-@EAT@EET0-@EASY@EEZ&-@E
ASE@.EZ-@ED1@33D-@EN@5N-@EX1@55KSPE9L#D[[0>6S-@EXAMPLE@3KZ95MP8L
-@EST@55ST-":GOSUB 350:GOTO 1270
1000 X$="@FOR@FOOR-@FIVE@FA;#&VV-@FULL@F%%L-@FROM@FR86MM-@FINE@F
;5#&N-@FRED@FR44DD-@FIRST@F/RST-@FINAL@F;#&N8LL-@FLASH@FL"+CHR$(
58)+">-@FIX@FI#KS-":GOSUB 350:GOTO 1270
1010 X$="@GO@G8OU-@GOES@G8[[UZZ0-@GOOD@G%%D-@GREAT@GR))*T-@GOD@G
A;DD-@GEORGE@DJ[[RDJ0-@GOT@G;0T-@GET@G55T-@GAD@G99D-@GAME@G@*#M-
@GARDEN@G;RD#N-":GOSUB 350:GOTO 1270
1020 X$="@HE@HEE0-@HELLO@H38L8[U-@HOW@H;[-@HARK@H;RK0-@HAD@H"+CH
R$(58)+"#D-@HAVE@H99VV-@HEAR@HER-@HIM@HI!#M-@HIS@HI!#Z-@HER@H/R-
@HAS@H"+CHR$(58)+"5X-@HURT@HH/RT-@HELEN@H538L#N-@HAND@H99ND-@HAP
PY@H99P&-":GOSUB 350:GOTO 1270
1030 X$="@I@;5#&-@IN@!!#N-@I'M@;#*M-@IT@!IT-@IS@I#ZZ-@IF@IF-@ING
@E+-@ICE@;5#&S-@INTERROGATE@I!#NT43RR[UG"+CHR$(94)+"T-@I'LL@;5#L
L-@IST@!ST-@IER@&&/-@IZE@A5#*Z-":GOSUB 350:GOTO 1270
1040 X$="@JEST@DJ43ST-@JUDGE@DJ777DJ-@JOHN@DJ;8NN-@JOAN@DJ8UUN-"
:GOSUB 350:GOTO 1270
```

Program continues

Program continued

```
1050 X$="@KICK@KK!!0K-@KNOW@N8"+CHR$(92)+"U-":GOSUB 350:GOTO 127
0
1060 X$="@LY@L&-@LEAF@L.F-@LINE@LA4&N-@LIGHT@L;#&T-@LIKE@L;#&0K-
@LIB@LI#B-@LONG@L,+-@LAMP@L"+CHR$(58)+"MP-@LESS@L3S-@LET@L43T-@L
AST@L"+CHR$(58)+"ST-@LESS1@L33S-@LARK@L;RK-@LIFE@LA5#F-@LIST@LIS
T-":GOSUB 350:GOTO 1270
1070 X$="@ME@MEE0-@MULTIPLY@M7LT8PL;&-@MY@M;5&-@MANY@MIN&-@MOUTH
@M;[0==0-@MASTER@M99ST/-@MOUTH2@M;[L<<0-@MORE@MO[R-@MAY@M5@Y0-@M
AN@M99NN-@MADE@M5@YDD-@MUST@M86ST-@MUCH@M86TC-@MOST@M8[UST-@MIGH
T@M;#&T-@MISS@MI#S-":GOSUB 350
1080 X$="@MENT@M55NT-@MOON@M('N-@MAKE@M5@YK0-@MUSIC@MY(X!0K-@MAN
UAL@M95NYU8L0-@MET@M43T-":GOSUB 350:GOTO 1270
1090 X$="@N@NN-@N'T@#NT-@NO@N8OU-@NINE@N;4&N-@NAME@N@*#M-@NOT@NA
A0T0-@NOW@N;U-@NEW@N((U-@NEXT@N35KST-@NESS@N4S-@NEVER@N43V/R-":G
OSUB 350:GOTO 1270
1100 X$="@ONE@W877NN-@ON@;ANN-@OF@66V-@OK@$[K@)*-@OUCH@AA[0TC0-@
O'CLOCK@%KL1K-@OR@O[R-@OUR@A[[/-@OUT@;OT-@OTHER@76</-@ONLY@OUNL&
-@OLD@8OLLDD0-@OWN@8OUN-@OPINE@"+CHR$(92)+"P;5#&N-@OVER@[[V/R-@O
UGHT@2270T-":GOSUB 350
1101 X$="@OPERATE@;P/@*T-":GOSUB 350:GOTO 1270
1110 X$="@PA1@ -@PA0@0-@PRAY@PR@*&-@PRONOUNCE@PRON;UNS-@PLAY@PL@
*&-@PIPER@P;#P/-@PICK@PI#K-@PECK@P445K-@PICKLED@PI#KLD0-@PETER@P
ET5/-@PEPPER@P3P/-@PILOT@P;5L5T-@PRISON@PR!#X!#N-@POEM@P8O4M-@PL
EASE@PL.EZ-@PRY@PR;5#&-":GOSUB 350
1120 X$="@PRACTICE@PR99KT!IS-@PENCIL@P54NS5L-@PLAIN@PL@*N-@PUT@P
%%T-@PROGRAM@PROUGR95M0-@PHONEME@F[[N*&M-@PROMISE@PR67M##S-":GOS
UB 350:GOTO 1270
1130 X$="@Q@00000-@QQ@0000000000-@QUICK@KWIK-@QUITE@KW;5#&T-":GO
SUB 350:GOTO 1270
1140 X$="@RUN@R86NN-@READY@R545D&-@ROSE@R8O#X-@RIB@RIB-@RESERVE@
REZ//VV-@RE@RE-@ROAM@R8UUM-@ROOM@RRUUM-":GOSUB 350:GOTO 1270
1150 X$="@S@S-@SAW@S21-@SPEECH@SP*&TC0-@SPEAK@SP*&K0-@SAID@S33D-
@SELL@S538L-@SIX@SI#K0S-@SEVEN@S54V4NN-@SUBTRACT@S7B0TR99K0T-@ST
OP@ST;8P-@SAVE@S5@&V-@SPOKE@SP[U0KK0-@SURE@>%%R-@SIR1@SS//-@SIR@
S/-@SHE@>EE-@SO@S8[U-@SOME@S66MM-@SHALL@>99L-":GOSUB 350
1160 X$="-@ST@0T0-@SUCH@S770TC-@SHOULD@>$D-@SERVE@S//V-@SCREEN@S
KR.N-@SHOW@>OO-@SHELL@>33L-@SEE@S.0-@SIGN@S;5#&N-@SIT@S"+CHR$(34
)+"T-@SAT@S"+CHR$(58)+"T-@SYNTHESIZE@SIN=8S;&Z-":GOSUB 350
1170 X$="@SPORT@SPORT-@SAY@S5)*&-@SECOND@S43K86ND-@SLEEP@SL.P-@S
IGH@S;5#&-@SIGHT@S;5#&T0-@STAR@ST;R-@SPOON@SP('N-@SOON@S('NN-@SO
N@S66N-@SUPER1@SUUP/0K"+CHR$(58)+"L#FR99DJ#L!ST!K-":GOSUB 350:GO
TO 1270
1180 X$="@T@DT-@THE@<<88-@TOP@T;AP-@TO@T(UU-@THAN@="+CHR$(58)+"N
-@THIS@<I#S-@TONGUE@T77+-@TIME@T;5#&M-@TWO@T((UU-@THREE@==R.&-@T
ALK@T227K0-@TRIP@TRIP-@THOUGHT@=2270T-@TROTH@00TR1#==-@TODAY@T(U
D@*&-@TOWN@T;88NN-@TAKE@T5@YK0-@TEN@T445N-":GOSUB 350
1181 X$="@THEY@<5)*&-":GOSUB 350
1190 X$="@THAT@="+CHR$(58)+"#T-@TOWER@TA[/R-@TURN@T//N-@THIRD@=/
/D-@THERE@<35#R-@THEN@<<35NN-@THEM@<44M-@THESE@<EEZ-@THOSE@<8[[U
Z-@THOUGH@<8"+CHR$(92)+"U-@THINK@=#E+K0-@TOGETHER@T(UGG35</R-@TR
Y@TR;5#&-@THANK@=99+K-@THING@=#!&+-":GOSUB 350:GOTO 1270
1200 X$="@UTTER@66T/-@UPON@6P%%N-@UP@67P-@US@66S-@UN@66N-@USE2@Y
('Z-@USE@Y('S-":GOSUB 350:GOTO 1270
1210 X$="@VIRGINIA@VV/DJI!N&8-@VERY@V45#RY-@VOICE@VO85&SS-":GOSU
B 350:GOTO 1270
1220 X$="@WAS@W6AX-@WINE@W;5#&N-@WILLARD@W!L/RD-@WILL@W"+CHR$(34
)+"L-@WHAT@WA#T-@WHERE@W45#R-@WE@W.0-@WHO@H((U-@WELL@W538L-@WHIC
H@WITC-@WERE@W/R-@WHEN@W44N-@WOULD@W$D-@WOMEN@WI#MI#NN-@WAIT@W@*
0T-@WORD@W/RD-":GOSUB 350
1230 X$="@WEEP@W.P-@WANT@W;NT-@WENT@W3#NT-@WORK@W/RK-@WON'T@W8ON
T-@WHOM@H((UMM-@WAY@W5@Y-@WITH@W"+CHR$(34)+"=-@WROTE@R8[[T-@WISE
@WA5#*Z-@WISE1@W;5#*Z-":GOSUB 350:GOTO 1270
1240 X$="@X-RAY@55KSS0R@*-":GOSUB 350:GOTO 1270
1250 X$="@Y@&-@YOU@Y'U-@YES@Y33S-@YOUR@&[[R-@YET@Y43T-@YE@YE.-":
GOSUB 350:GOTO 1270
1260 X$="@Z@Z-@Z4@L0L0L0L0L0L0L0L0-@ZERO@Z*#R8[U-@ZEAL@ZEEL-@Z
IP@ZIP-@ZOOKS@Z(UKK0S-@ZOOM@Z(UM-":GOSUB 350
1270 X$="13":GOSUB 350 :'-----"13" IS UNLUCKY-WORD IS MISSING.
1280 STOP
1290 '-----------------------------------------------------------
1300 '-----WHAT FOLLOWS IS A MINI-PROGRAM THAT HELPS IN THE
1310 '-----SELECTION OF PHONEMES FOR A NEW WORD. . .
1320 '-----TO ACTIVATE:    RUN 1330
1330 CLS:PRINT "PUT TRIAL PHONEMES IN T$, LINE 1360"
1340 PRINT "TO HEAR PHONEMES:   RUN 1350":STOP
1350 CLS
1360 T$="77DI>8N"
1370 FOR N=1 TO 4
1380 PRINT@ 992, "? "+T$+" ?";
1390 FOR T=1 TO 900:NEXT T,N
1400 PRINT@ 768, "T$ IS IN LINE 1360"
1410 PRINT "(RUN 1350)"
1420 PRINT "PHONEMES USED WERE:    ";T$
1430 STOP
```

THE PROGRAM STORE

Crush, Crumble and Chomp!

From Epyx

It's a monster movie, and you are the monster! This action game is loaded with graphics and sound as you practice your villany. With 6 monsters, 4 cities, and 5 game objectives, you get a choice of more than 100 possible scenarios. A monster's life is not all carnivorous crunching, though: The combined resources of the police, science, and armed forces are bent on your destruction. It's a struggle of might and strategy all the way.

16K tape...$29.95 32K disk...$29.95

Empire of the Over-Mind

By N.M.A. from Avalon Hill

A fantasy adventure where you are summoned by good King Alcazar to defeat the tyrannical Over-Mind and reclaim the kingdoms of the red and blue planets.

Throughout your journey, you are guided by the epic "Rhyme of Over-Mind," with its characterizations and clues. The Over-Mind is part machine, part evil force -- a most devastating opponent, indeed. Good luck!

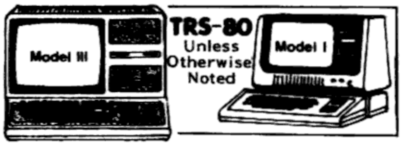
Tape for 40K Atari,
48K Apple & TRS-80...$30.00

By John Allen from Acorn

More features, thrills, and sound than even John Allen's famous PINBALL. Once you load ASTROBALL into your TRS-80, the arrow keys become flipper buttons, the screen becomes the play board, and you become the "Pinball Wizard!"

A flying saucer, spaceships, meteors, and black holes add to the fun as your ball realistically zings around the board. ASTROBALL will have all your family and friends lining up for the pinball action and challenge. Five skill levels.

16K protected tape...$19.95
32K protected disk...$19.95



GAUNTLET OF DEATH

From Programmer's Guild

You can almost hear the "C-L-A-N-K!" of the doors shutting behind you as you enter the gauntlet. Before you stretch graphically depicted corridors, leading to the rescue of Chief Broton's daughter and safe exit -- or to a hideous death.

Spiders, poisoned darts, and other surprises haunt the halls, along with magic potions and useful treasures. Skill and strategy will help, but luck and determination are needed to successfully run the gauntlet!

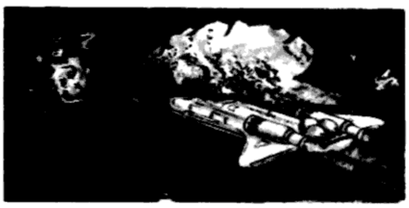
16K tape...$19.95

MIND THRUST

By Sackson & Wazaney from Hayden

Match wits with the computer in this deceptively simple game. Your object is to complete an unbroken chain across a 6 X 8 gameboard grid. On each turn you may either place a new "link" on your chain or remove one from the computer's chain. Sound easy? Just wait until you try it!

16K tape...$16.95



SPACE ROCKS

By Steven Kearns from Acorn

Gigantic antimatter rocks appear on the Tactical Display Screen of your spacecraft. You blast away with lasers and they just explode into smaller chunks for you to destroy. To add to your woes, time bombs appear periodically. If their timers reach zero -- BOOM! And if that's not enough, the aliens will be glad to send out some spaceships loaded with antimatter torpedoes. Fire thrusters to move, shoot laser cannon, jump to hyperspace -- anything to avoid the onslaught. One or two players can compete, with five levels of difficulty.

16K protected tape..$19.95
32K protected disk..$19.95

ROBOT ATTACK

By Hogue & Konyu from Big Five

One of the top names in TRS-80 arcade games adds a new dimension: voice sound effects! You have to be quick to keep your head on straight in this "search and destroy" arcade game. The innovations built into ROBOT ATTACK take your TRS-80 near the limits of its capabilities. You MUST see and hear it!

16K tape...$15.95 32K disk...$19.95

MANAGEMENT DECISIONS

By Robert Montgomery from Hayden

An incredibly detailed and complex simulation of business management. You are given the reins of a large business, controlling every aspect of its operation for 20 quarters (five years) or until you resign.

Many reports, charts, and graphs are available to keep you abreast of happenings both internal and external to your company. Sales, advertising, production, competition, and economic conditions all interrelate as you try to earn profits for yourself -- and your stockholders. This program is so informative that we think it should grant you credits in Business Administration!

16K tape, Model I or III...$49.95
32K disk, Model I only.....$54.95

Honestly...

BASIC COMPILERS

BASIC compilers may interest you because compiled programs may run many times faster than regular BASIC.

The ideal compiler would take any BASIC program and compile it directly to machine language. The difficulty lies in the "trick" features written into many programs, like string packing, sound effects, etc. Other problems include non-standard structures like breaking out of a FOR...NEXT loop or a subroutine. All compilers may require modifications to your BASIC program. We have found ACCEL 2 to require the least. BASIC PROGRAMMING ASSISTANT (model I, $14.95) is useful in finding FOR...NEXT loops and modifying programs; PACKER ($29.95) in some cases will make a program compilable.

	Allen Gelder's ACCEL 2	Simultek's ZBASIC	Microsoft's BASCOM
Minimum Hardware	16K RAM Tape or disk	16K RAM Tape or disk	32K RAM Disk only
Model III compatible	YES	NO	NO
Optimal memory utilization	YES	NO	NO
All BASIC instructions	YES	NO	YES
All variable types & floating point	YES	NO	YES
Support I/O for tape	YES	NO	NO
Unrestricted commercial use	YES	YES	NO

ZBASIC requires too many modifications to your BASIC program in almost every case. Microsoft's BASCOM is the easiest compiler to use if you have disk drives. However, it is more expensive, doesn't support string packing, and requires more memory. We recommend ACCEL 2 because it will work with models I or III, requires a minimum amount of memory, and will work with most BASIC programs.

ZBASIC from Simultek: Tape version.......$79.95
ZBASIC from Simultek: Disk version.......$89.95
BASCOM from Microsoft: Disk only........$149.95
ACCEL 2 from Allen Gelder: Tape & disk...$88.95
(To save ACCEL-compiled programs to tape you also need TSAVE, $9.95)

Visit our other stores: Seven Corners Center · Falls Church, VA & W. Bell Plaza · 6600 Security Blvd. · Baltimore, MD

TO ORDER CALL TOLL FREE 800 424-2738 For information Call (202) 363-9797

THE PROGRAM STORE
4200 Wisconsin Avenue NW, Dept. E10 Box 9609
Washington, D.C. 20016

✓17

MAIL ORDERS: Send check or M.O. for total purchase price, plus $1.00 postage & handling. D.C. residents, add 6% tax. Charge card customers: include all embossed information on card.

See List of Advertisers on page 386

Programming intelligence into your TRS-80.

Emterm

Photo 1. Lynx Telephone Linkage System

Program Listing

```
4590          00100          ORG     4590H
              00110 ;**********RELOCATION ROUTINE**********
4590 CD5A4B   00120 RELSRT: CALL    SCHTCO          ;CLEAR SCREE
N
4593 21F846   00130 RELCON: LD      HL,MSG1A        ;NEW ADDR QU
ESTION
4596 CDA728   00140         CALL    28A7H
4599 21D946   00150         LD      HL,MSHD         ;HL=MS ADDR
SAVE POINTER
```

Program continues

Jeffery A. Mills
Emtrol Systems, Inc.
123 Locust Street
Lancaster, PA 17602

If you have been thinking of using your TRS-80 as an intelligent terminal, the program presented here will interest you. It was written to support the Lynx telephone linkage system and will also work with a Radio Shack RS232 interface. Features include: relocation, message handling, Basic transmission and reception, software-controlled UART programming, intelligent terminal operation, and return to Basic capability.

Program Operation

Most Emterm commands are entered in response to a menu display or a prompt on the CRT. Some commands are always active, and can be entered at any time. The following is a complete list of Emterm commands and actions:

Store Message (S)-Causes display of mini-menu:
Store Message (S).
Erase Message (E).
S-Enables storage of up to 1039 keyboard characters in the temporary buffer.
E- Erases the temporary buffer.
Transparent commands—Shift ↑ causes a return to the monitor menu, Shift @ causes a return to the mini-menu.

Receive Basic (R)-Causes display of: "Receiving Program," and a blinking asterisk after the AAA sync character is received.
Transparent commands—Shift ↑ causes a return to the monitor menu.

Transmit Basic (X)-Causes display of the mini-menu:
Transmit Basic (T).
Load Program (L).
T-Transmit Basic program stored in

the Basic buffer. Sends out a AAA sync character before the program.

L-Prompts the operator to ready the cassette and hit L to load.

Transparent commands-Shift ↑ causes a return to the monitor menu. Note: This is disabled during Basic transmission.

Terminal (T)-Enables intelligent terminal operation.

Transparent commands-Shift ↑ causes a return to the monitor menu. Shift → causes text stored in the temporary message buffer to be transmitted. Shift → enables/disables automatic error display. Shift ← enables/disables the parallel printer operation. Break transmits a string of zeros. Clear cleans the screen. Note: Upon entry to the terminal mode, the automatic error display is enabled and the printer output is disabled.

View/Change UART Configuration-Permits viewing of the UART configuration and change under software control as follows:

Parity: Odd/Even/None;
Word Length: 5/6/7/8;
Stop Bits: 1 or 2.

In response to change question, operator can enter N or Enter for no, Y for yes. Transparent commands-Shift ↑ causes a return to the monitor menu.

Back to Basic (B)-Causes a return to Level II Basic.

Program Details

The program consists of seven routines, with the following functions:

- Relocator Routine-When first loaded, the program occupies memory locations 470FH to 4E34H. A relocation offset is determined by subtracting 470FH from the new starting address entered by the user. The resulting value is added to the address bytes of all instructions that must be modified for relocation. When this change is completed, the modified code is transferred to the new area of memory, and program execution begins at the new starting address.
- Store Message Routine-This reserves 1039 bytes of memory below the main program for a message storage buffer. This area is initialized to all zeros on program start-up. When the message storage routine is executed, the buffer is scanned for a zero byte to indicate free storage area. Any stored text is displayed in the process. If space exists for additional message text, the cursor stops and text may be entered. If the end of the buffer is reached at any time, a full-buffer message is displayed.
- Transmit Message Routine-This routine causes a scan of the message buffer for a zero byte. All text encountered during

"Most Emterm commands are entered in response to a menu display or prompt. . ."

Program continued

```
459C 0604     00160         LD      B,04H         ;B=ADDR DIGI
T COUNT
459E CD2B00   00170 WFKE:   CALL    002BH         ;SCAN KEYBOA
RD
45A1 B7       00180         OR      A
45A2 28FA     00190         JR      Z,WFKE        ;WAIT FOR EN
TRY
45A4 FE08     00200         CP      08H           ;BACKSPACE E
NTERED?
45A6 200E     00210         JR      NZ,CFHD       ;IF NOT,CHEC
K HEX ENTRY
45A8 3E04     00220         LD      A,04H         ;COMPLETE AD
DR ENTERED?
45AA B8       00230         CP      B
45AB 28F1     00240         JR      Z,WFKE        ;IF NOT, WAI
T FOR MORE
45AD 04       00250 INCB:   INC     B             ;ADJUST DIGI
T COUNT
45AE 3E08     00260         LD      A,08H         ;DELETE LAST
 ON CRT
45B0 CD3300   00270         CALL    0033H
45B3 2B       00280         DEC     HL            ;ADJUST ADDR
 SAVE POINTER
45B4 18E8     00290         JR      WFKE          ;CONTINUE KE
YBOARD SCAN
45B6 FE30     00300 CFHD:   CP      30H           ;VALID HEX E
NTRY?
45B8 3F       00310         CCF
45B9 D2CE45   00320         JP      NC,BADNUM     ;IF NOT, DON
'T ACCEPT
45BC FE3A     00330         CP      3AH           ;CHECK FOR 0
-9 ENTRY
45BE 3F       00340         CCF
45BF D2D945   00350         JP      NC,GZTN       ;SAVE 0-9 EN
TRY
45C2 FE41     00360         CP      41H           ;CHECK FOR I
LLEGAL ENTRY
45C4 3F       00370         CCF
45C5 D2CE45   00380         JP      NC,BADNUM     ;BAD ENTRY,
DON'T ACCEPT
45C8 FE47     00390         CP      47H           ;CHECK FOR A
-F ENTRY
45CA 3F       00400         CCF
45CB D2D045   00410         JP      NC,GATF       ;SAVE A-F EN
TRY
45CE 18CE     00420 BADNUM: JR      WFKE          ;CONTINUE SC
ANNING KEYS
45D0 CD3300   00430 GATF:   CALL    0033H         ;DISPLAY A-F
 ADDR DIGIT
45D3 E60F     00440         AND     0FH           ;A-F ASCII T
O HEX CONV
45D5 C609     00450         ADD     A,09H
45D7 1805     00460         JR      SAVEHN        ;SAVE HEX RE
SULT
45D9 CD3300   00470 GZTN:   CALL    0033H         ;DISPLAY 0-9
 ADDR DIGIT
45DC E60F     00480         AND     0FH           ;0-9 ASCII T
O HEX CONV
45DE 77       00490 SAVEHN: LD      (HL),A        ;SAVE HEX RE
SULT
45DF 23       00500         INC     HL            ;INC ADDR SA
VE POINTER
45E0 10BC     00510         DJNZ    WFKE          ;GET REST OF
 NEW ADDR
45E2 CD2B00   00520 WFEK:   CALL    002BH         ;SCAN KEYS
45E5 B7       00530         OR      A
45E6 28FA     00540         JR      Z,WFEK        ;WAIT FOR EN
TRY
45E8 FE0D     00550         CP      0DH           ;CR ENTERED?
45EA 2806     00560         JR      Z,PHN         ;IF SO, PACK
 HEX NIBBLES
45EC FE08     00570         CP      08H           ;BACKSPACE E
NTERED?
45EE 20F2     00580         JR      NZ,WFEK       ;IF NOT, SCA
```

Program continues

"A full buffer message is displayed after the last character is transmitted."

the scan is sent to the UART for transmission. If the entire buffer is full, a full-buffer message is displayed after the last character is transmitted.

● Basic Transmission Routine–Basic programs are stored in a compressed code format, starting at 42E9H. Programs can be entered from the keyboard in Level II Basic or from cassette tape. This routine simply dumps the Basic buffer, starting at 42E9H, and continues until the end of the program, indicated by three zeros in sequence. A three A sync character is sent out for the receiving TRS-80 just before program transmission begins.

● Basic Reception Routine–This routine monitors the UART for a three A sync character. Incoming program bytes are stored in the Basic buffer starting at 42E9H. Three zeros in sequence indicate the end of the program.

● Terminal Routine–This routine controls UART operation. When a character becomes available, the UART receive buffer is read and the character is transferred to the display. When you make a keyboard entry, the UART transmit buffer is polled until empty, at which time the keyboard entry is transmitted.

● View/Change UART Configuration Routine–During program start-up hardware switches which contain UART programming data are read and stored in memory. Each time the UART is reconfigured, it is from this byte in memory that the configuration information is taken. The View/Change routine allows the user to read and modify this location under software control. ■

Emterm is available on cassette from Emtrol Systems, Inc.

Program continued

```
N KEYS
45F0 18BB      00590        JR      INCB          ;DELETE LAST
, GET ANOTHER
45F2 21D946    00600 PHN:   LD      HL,MSHD       ;SET UP ADDR
 POINTER
45F5 AF        00610        XOR     A             ;CLEAR ACC
45F6 ED6F      00620        RLD                   ;XCHANGE NIB
BLES (HL)<>ACC
45F8 7E        00630        LD      A,(HL)        ;GET MS ADDR
 NIBBLE
45F9 23        00640        INC     HL            ;INC ADDR PO
INTER
45FA B6        00650        OR      (HL)          ;PACK MS ADD
R BYTE
45FB 32D946    00660        LD      (MSHD),A      ;SAVE MS ADD
R BYTE
45FE 23        00670        INC     HL            ;INC ADDR PO
INTER
45FF AF        00680        XOR     A             ;CLEAR ACC
4600 ED6F      00690        RLD                   ;XCHANGE NIB
BLES (HL)<>ACC
4602 7E        00700        LD      A,(HL)        ;GET LS ADDR
 NIBBLE
4603 23        00710        INC     HL            ;INC ADDR PO
INTER
4604 B6        00720        OR      (HL)          ;PACK LS ADD
R BYTE
4605 32DA46    00730        LD      (MSHD+1),A    ;SAVE LS ADD
R BYTE
4608 21D946    00740        LD      HL,MSHD       ;SET UP ADDR
 POINTER
460B 7E        00750        LD      A,(HL)        ;NEW ADDR >
470F?
460C FE47      00760        CP      47H
460E 3F        00770        CCF
460F D2DF46    00780        JP      NC,ERRORA     ;ERROR, MS A
DDR BYTE <47
4612 2802      00790        JR      Z,CLSBS       ;MS=47, CHEC
K LS BYTE
4614 180C      00800        JR      CUE           ;NEW ADDR <
F8DA?
4616 23        00810 CLSBS: INC     HL            ;INC ADDR PO
INTER
4617 7E        00820        LD      A,(HL)        ;GET LS ADDR
 BYTE
4618 FE0F      00830        CP      0FH           ;>0F?
461A 3F        00840        CCF
461B 2B        00850        DEC     HL            ;DEC ADDR PO
INTER
461C DA3546    00860        JP      C,COFST       ;IF LS ADDR
OK, GET OFFSET
461F C3DF46    00870        JP      ERRORA        ;IF LS ADDR
BAD, ERROR
4622 FEF8      00880 CUE:   CP      0F8H          ;CHECK HIGH
ADDR
4624 3F        00890        CCF
4625 2805      00900        JR      Z,CLSB        ;IF MS OK, C
HECK LS BYTE
4627 DADF46    00910        JP      C,ERRORA      ;IF MS BAD,
ERROR
462A 1809      00920        JR      COFST         ;COMPUTE OFF
SET
462C 23        00930 CLSB:  INC     HL            ;INC ADDR PO
INTER
462D 7E        00940        LD      A,(HL)        ;GET LS ADDR
 BYTE
462E FEDB      00950        CP      0DBH          ;CHECK LS BY
TE
4630 3F        00960        CCF
4631 DADF46    00970        JP      C,ERRORA      ;IF BAD, ERR
OR
4634 2B        00980        DEC     HL            ;DEC ADDR PO
INTER
4635 56        00990 COFST: LD      D,(HL)        ;COMPUTE OFF
SET BETWEEN
```

Program continues

Program continued

```
4636 23       01000          INC    HL             ;NEW ADDR AN
D OLD ADDR
4637 5E       01010          LD     E,(HL)
4638 AF       01020          XOR    A              ;CLEAR ACC
4639 210F47   01030          LD     HL,470FH       ;LOAD OLD AD
DR
463C EB       01040          EX     DE,HL
463D ED52     01050          SBC    HL,DE          ;PERFORM SUB
TRACTION
463F 22DD46   01060          LD     (OFFSET),HL    ;SAVE OFFSET
4642 210F47   01070          LD     HL,470FH       ;SET UP ADDR
 POINTER
4645 7E       01080 EMB:     LD     A,(HL)         ;GET PROGRAM
 BYTE
4646 FE21     01090          CP     21H            ;CHECK FOR O
P CODES WITH
4648 2829     01100          JR     Z,IBP1         ;ADDRS THAT
MUST BE
464A FE11     01110          CP     11H            ;MODIFIED, A
ND JUMP TO
464C 2825     01120          JR     Z,IBP1         ;RELOCATION
ROUTINE WHEN
464E FE32     01130          CP     32H            ;FOUND
4650 2821     01140          JR     Z,IBP1
4652 FE3A     01150          CP     3AH
4654 281D     01160          JR     Z,IBP1
4656 FECD     01170          CP     0CDH
4658 200B     01180          JR     NZ,TFCA
465A 23       01190          INC    HL             ;INC ADDR PO
INTER
465B 7E       01200          LD     A,(HL)         ;GET PROGRAM
 BYTE
465C FECD     01210          CP     0CDH           ;=CD?
465E 2803     01220          JR     Z,NOC          ;IF SO, ADJ
ADDR POINTER
4660 2B       01230          DEC    HL             ;DEC ADDR PO
INTER
4661 1810     01240          JR     IBP1           ;TO RELOCATI
ON ROUTINE
4663 2B       01250 NOC:     DEC    HL             ;DEC ADDR PO
INTER
4664 7E       01260          LD     A,(HL)         ;GET PROGRAM
 BYTE
4665 FECA     01270 TFCA:    CP     0CAH           ;=CA?
4667 280A     01280          JR     Z,IBP1         ;IF SO, RELO
CATE
4669 FEC3     01290          CP     0C3H           ;=C3?
466B 2806     01300          JR     Z,IBP1         ;IF SO, RELO
CATE
466D FEC4     01310          CP     0C4H           ;=C4?
466F 2802     01320          JR     Z,IBP1         ;IF SO, RELO
CATE
4671 182A     01330          JR     IBP2           ;FINISHED RE
LOCATING?
4673 23       01340 IBP1:    INC    HL             ;INC ADDR PO
INTER
4674 23       01350          INC    HL             ;INC ADDR PO
INTER
4675 7E       01360          LD     A,(HL)         ;GET MS ADDR
 BYTE
4676 FE43     01370          CP     43H            ;RELOCATE IF
 >43
4678 3F       01380          CCF
4679 D28946   01390          JP     NC,CNB         ;OTHERWISE,
CHECK NEXT
467C E6F0     01400          AND    0F0H           ;ISOLATE MS
NIBBLE
467E FE40     01410          CP     40H            ;IGNORE IF N
OT=4X
4680 2802     01420          JR     Z,CSFADR       ;IF=4X, RELO
CATE
4682 1805     01430          JR     CNB            ;CHECK NEXT
BYTE
4684 CD8C46   01440 CSFADR:  CALL   SFADR          ;CALL RELOCA
```

Program continues

✓See List of Advertisers on page 386

Program continued

```
TION SUB
4687 1814      01450         JR     IBP2          ;FINISHED RE
LOCATING?
4689 2B        01460 CNB:    DEC    HL            ;DEC ADDR PO
INTER
468A 18B9      01470         JR     EMB           ;CHECK IF RE
LO NECESSARY
468C ED5BDD46  01480 SFADR:  LD     DE,(OFFSET)   ;ADDR CHANGE
 SUBROUTINE
4690 E5        01490         PUSH   HL            ;SAVE ADDR P
OINTER
4691 46        01500         LD     B,(HL)        ;ADD OFFSET
TO PRESENT
4692 2B        01510         DEC    HL            ;PROGRAM ADD
R
4693 4E        01520         LD     C,(HL)
4694 EB        01530         EX     DE,HL
4695 09        01540         ADD    HL,BC         ;PERFORM ADD
ITION
4696 EB        01550         EX     DE,HL
4697 E1        01560         POP    HL            ;RETRIEVE AD
DR POINTER
4698 72        01570         LD     (HL),D        ;LOAD NEW AD
DR
4699 2B        01580         DEC    HL
469A 73        01590         LD     (HL),E
469B 23        01600         INC    HL
469C C9        01610         RET
469D 23        01620 IBP2:   INC    HL            ;INC ADDR PO
INTER
469E 3E4E      01630         LD     A,4EH         ;FINISHED RE
LOCATING?
46A0 BC        01640         CP     H
46A1 2802      01650         JR     Z,CLSB1       ;CHECK LSD A
DDR BYTE
46A3 18A0      01660         JR     EMB           ;IF MS BYTE
= 4E
46A5 3E35      01670 CLSB1:  LD     A,35H
46A7 BD        01680         CP     L             ;DONE IF LS
= 35
46A8 2802      01690         JR     Z,FRB         ;IF SO, GO T
O NEW ADDR
46AA 1899      01700         JR     EMB           ;CONTINUE RE
LOCATION
46AC 211C48    01710 FRB:    LD     HL,481CH      ;RELOCATE SP
ECIAL OP CODES
46AF CD8C46    01720         CALL   SFADR
46B2 21D94B    01730         LD     HL,4BD9H
46B5 CD8C46    01740         CALL   SFADR
46B8 ED5BDD46  01750         LD     DE,(OFFSET)   ;GET OFFSET
46BC 210F47    01760         LD     HL,470FH      ;DETERMINE N
EW STARTING
46BF 19        01770         ADD    HL,DE         ;ADDR
46C0 112507    01780         LD     DE,0725H      ;LOAD PROGRA
M LENGTH
46C3 19        01790         ADD    HL,DE         ;DETERMINE N
EW ENDING ADDR
46C4 EB        01800         EX     DE,HL         ;SET UP FOR
BLOCK TRANSFER
46C5 012607    01810         LD     BC,0726H
46C8 21344E    01820         LD     HL,4E34H
46CB EDB8      01830         LDDR                 ;PERFORM BLO
CK TRANSFER
46CD 3AD946    01840         LD     A,(MSHD)      ;GET NEW STA
RTING
46D0 67        01850         LD     H,A           ;ADDR
46D1 3ADA46    01860         LD     A,(MSHD+1)
46D4 6F        01870         LD     L,A
46D5 22DF40    01880         LD     (40DFH),HL
46D8 E9        01890         JP     (HL)          ;EXECUTE PRO
GRAM
0004           01900 MSHD:   DEFS   4             ;SET UP STOR
AGE
0002           01910 OFFSET: DEFS   2
46DF 21E846    01920 ERRORA: LD     HL,MSG2A      ;ERROR MESSA
GE
46E2 CDA728    01930         CALL   28A7H
46E5 C39345    01940         JP     RELCON        ;BACK TO REL
OCATOR
46E8 0D        01950 MSG2A:  DEFB   0DH           ;MESSAGE TAB
LE
46E9 49        01960         DEFM   'ILLEGAL ENTRY'
46F6 0D        01970         DEFB   0DH
46F7 00        01980         DEFB   00H
```

Program continues

✓See List of Advertisers on page 386

Program continued

```
46F8 4E        Ø199Ø MSG1A:  DEFM    'NEW STARTING ADDRESS?'
47ØD ØD        Ø2ØØØ         DEFB    ØDH
47ØE ØØ        Ø2Ø1Ø         DEFB    ØØH
47ØF           Ø2Ø2Ø         ORG     47ØFH
               Ø2Ø3Ø ;*********TERMINAL PROGRAM**********
47ØF 3EC9      Ø2Ø4Ø EMTERM: LD      A,ØC9H           ;LOAD RET IN
ST OP CODE
4711 32ØC4Ø    Ø2Ø5Ø         LD      (4ØØCH),A        ;STORE RET I
NSTRUCTIONS
4714 32ØF4Ø    Ø2Ø6Ø         LD      (4ØØFH),A        ;IN LII RAM
AREA
4717 32124Ø    Ø2Ø7Ø         LD      (4Ø12H),A
471A 32134Ø    Ø2Ø8Ø         LD      (4Ø13H),A
471D 32354Ø    Ø2Ø9Ø         LD      (4Ø35H),A
472Ø 328D4Ø    Ø21ØØ         LD      (4Ø8DH),A
4723 32954Ø    Ø211Ø         LD      (4Ø95H),A
4726 32984Ø    Ø212Ø         LD      (4Ø98H),A
4729 21A641    Ø213Ø         LD      HL,41A6H
472C Ø615      Ø214Ø         LD      B,15H
472E 77        Ø215Ø LOOP:   LD      (HL),A
472F 23        Ø216Ø         INC     HL
473Ø 23        Ø217Ø         INC     HL
4731 23        Ø218Ø         INC     HL
4732 1ØFA      Ø219Ø         DJNZ    LOOP             ;CONTINUE LO
ADING RETS
4734 21ØØ43    Ø22ØØ         LD      HL,EMTERM-4ØFH   ;MESSAGE BUF
FER START
4737 AF        Ø221Ø KOCM:   XOR     A                ;CLEAR ACC
4738 77        Ø222Ø         LD      (HL),A           ;ZERO BUFFER
 BYTE
4739 23        Ø223Ø         INC     HL               ;POINT TO NE
XT BYTE
473A E5        Ø224Ø         PUSH    HL               ;SAVE POINTE
R
473B 11ØF47    Ø225Ø         LD      DE,EMTERM        ;SEE IF AT E
ND OF BUFFER
473E EB        Ø226Ø         EX      DE,HL
473F AF        Ø227Ø         XOR     A
474Ø ED52      Ø228Ø         SBC     HL,DE            ;SUBTRACT PO
INTER FROM END
4742 28Ø3      Ø229Ø         JR      Z,PHL            ;BUFFER FULL
 IF Ø RESULT
4744 E1        Ø23ØØ         POP     HL               ;RETRIEVE BU
FFER POINTER
4745 18FØ      Ø231Ø         JR      KOCM             ;KEEP ZEROIN
G BUFFER
4747 E1        Ø232Ø PHL:    POP     HL               ;RETRIEVE BU
FFER POINTER
4748 D3E8      Ø233Ø         OUT     (ØE8H),A         ;RESET UART
474A DBE9      Ø234Ø         IN      A,(ØE9H)         ;READ CONFIG
 SWITCHES
474C 32CF4D    Ø235Ø         LD      (HUCR),A         ;SAVE SWITCH
 READING
474F CD9B4B    Ø236Ø         CALL    SRAPU            ;RESET AND P
ROGRAM UART
4752 CD5A4B    Ø237Ø         CALL    SCHTCO           ;CLEAR SCREE
N
4755 21ØC4C    Ø238Ø         LD      HL,MSGØ          ;DISPLAY EMT
ROL MESSAGE
4758 CDA728    Ø239Ø         CALL    28A7H
475B 21DD4C    Ø24ØØ         LD      HL,MSG2Ø         ;DISPLAY MEN
U
475E CDA728    Ø241Ø         CALL    28A7H
4761 18ØC      Ø242Ø         JR      WFMK             ;WAIT FOR MO
NITOR KEY
4763 CD5A4B    Ø243Ø MON:    CALL    SCHTCO           ;CLEAR SCREE
N
4766 CD9B4B    Ø244Ø         CALL    SRAPU            ;RESET AND P
ROGRAM UART
4769 21CB4C    Ø245Ø         LD      HL,MSG1Ø         ;DISPLAY LYN
X MESSAGE
476C CDA728    Ø246Ø         CALL    28A7H
476F CD2BØØ    Ø247Ø WFMK:   CALL    ØØ2BH            ;SCAN KEYS
4772 B7        Ø248Ø         OR      A
4773 28FA      Ø249Ø         JR      Z,WFMK           ;WAIT FOR EN
TRY
4775 FE53      Ø25ØØ         CP      'S'              ;IF S, TO ST
ORE MESSAGE
4777 2822      Ø251Ø         JR      Z,STORE          ;ROUTINE
4779 FE52      Ø252Ø         CP      'R'              ;IF R, TO RE
CEIVE BASIC
477B CA8348    Ø253Ø         JP      Z,REC            ;ROUTINE
477E FE58      Ø254Ø         CP      'X'              ;IF X, TO TR
ANSMIT BASIC
```

Program continues

Program continued

```
 4780 CAE948    Ø255Ø          JP     Z,XMIT         ;ROUTINE
 4783 FE54      Ø256Ø          CP     'T'            ;IF T, TO TE
RMINAL ROUTINE
 4785 CAA549    Ø257Ø          JP     Z,TERM
 4788 FE56      Ø258Ø          CP     'V'            ;IF V, TO VI
EW/CHANGE UART
 478A CA4A4A    Ø259Ø          JP     Z,VIEW         ;CONFIGURATI
ON
 478D FE42      Ø26ØØ          CP     'B'            ;IF B, RETUR
N TO BASIC
 478F CACCØ6    Ø261Ø          JP     Z,Ø6CCH
 4792 FE1B      Ø262Ø          CP     1BH            ;IF SHIFT UP
 ARROW, RETURN
 4794 28CD      Ø263Ø          JR     Z,MON          ;TO MONITOR
MENU
 4796 CD944B    Ø264Ø          CALL   SERROR         ;DISPLAY ERR
OR MESSAGE
 4799 18D4      Ø265Ø          JR     WFMK           ;WAIT FOR MO
NITOR KEY
 479B CD5A4B    Ø266Ø STORE:   CALL   SCHTCO         ;CLEAR SCREE
N
 479E 21EB4D    Ø267Ø STORE1:  LD     HL,MSG18       ;DISPLAY MES
SAGE MENU
 47A1 CDA728    Ø268Ø          CALL   28A7H
 47A4 CD2BØØ    Ø269Ø GCTS:    CALL   ØØ2BH          ;SCAN KEYS
 47A7 B7        Ø27ØØ          OR     A
 47A8 28FA      Ø271Ø          JR     Z,GCTS         ;WAIT FOR EN
TRY
 47AA FE53      Ø272Ø          CP     'S'            ;IF S, TO ST
ORE ROUTINE
 47AC 2812      Ø273Ø          JR     Z,STO
 47AE FE45      Ø274Ø          CP     'E'            ;IF E, TO ER
ASE ROUTINE
 47BØ CA6F48    Ø275Ø          JP     Z,ERA
 47B3 FE1B      Ø276Ø          CP     1BH            ;IF SHIFT UP
 ARROW, TO
 47B5 28AC      Ø277Ø          JR     Z,MON          ;MONITOR MEN
U
 47B7 FE6Ø      Ø278Ø          CP     6ØH            ;IF SHIFT @,
 TO MESSAGE
 47B9 28EØ      Ø279Ø          JR     Z,STORE        ;MENU
 47BB CD944B    Ø28ØØ          CALL   SERROR         ;DISPLAY ERR
OR MESSAGE
 47BE 18E4      Ø281Ø          JR     GCTS           ;GET ANOTHER
 ENTRY
 47CØ CD5A4B    Ø282Ø STO:     CALL   SCHTCO         ;CLEAR SCREE
N
 47C3 21ØØ43    Ø283Ø          LD     HL,EMTERM-4ØFH ;POINT TO BU
FFER START
 47C6 7E        Ø284Ø          LD     A,(HL)         ;GET BUFFER
BYTE
 47C7 FEØØ      Ø285Ø          CP     ØØH            ;IF Ø, READY
 FOR MORE TEXT
 47C9 2833      Ø286Ø          JR     Z,CM
 47CB CD33ØØ    Ø287Ø DMB:     CALL   ØØ33H          ;DISPLAY MES
SAGE CHARACTER
 47CE CD2BØØ    Ø288Ø          CALL   ØØ2BH          ;SCAN KEYS
 47D1 B7        Ø289Ø          OR     A
 47D2 28Ø8      Ø29ØØ          JR     Z,CON1         ;NO ENTRY, C
ONTINUE
 47D4 FE6Ø      Ø291Ø          CP     6ØH            ;IF SHIFT @,
 TO MESSAGE
 47D6 28C3      Ø292Ø          JR     Z,STORE        ;MENU
 47D8 FE1B      Ø293Ø          CP     1BH            ;IF SHIFT UP
 ARROW, TO
 47DA 2887      Ø294Ø          JR     Z,MON          ;MONITOR MEN
U
 47DC 23        Ø295Ø CON1:    INC    HL             ;POINT TO NE
XT BUFFER BYTE
 47DD F5        Ø296Ø          PUSH   AF             ;SAVE CHARAC
TER
 47DE E5        Ø297Ø          PUSH   HL             ;SAVE POINTE
R
 47DF 11ØF47    Ø298Ø          LD     DE,EMTERM      ;SEE IF BUFF
ER FULL
 47E2 EB        Ø299Ø          EX     DE,HL
 47E3 AF        Ø3ØØØ          XOR    A              ;CLEAR ACC
 47E4 ED52      Ø3Ø1Ø          SBC    HL,DE          ;SUBTRACT PO
INTER FROM
 47E6 28Ø9      Ø3Ø2Ø          JR     Z,BFM          ;END OF BUFF
ER
 47E8 E1        Ø3Ø3Ø          POP    HL             ;RETRIEVE PO
INTER
```

Program continues

Program continued

```
47E9 F1         03040          POP     AF             ;RETRIEVE CH
ARACTER
47EA 7E         03050          LD      A,(HL)         ;GET CHARACT
ER  INTO ACC
47EB FE00       03060          CP      00H            ;IF 0, CONTI
NUE ENTERING
47ED 280F       03070          JR      Z,CM           ;MESSAGE
47EF 18DA       03080          JR      DMB            ;DISPLAY CHA
RACTER
47F1 E1         03090 BFM:     POP     HL             ;RETRIEVE PO
INTER
47F2 F1         03100          POP     AF             ;RETRIEVE CH
ARACTER
47F3 CD5A4B     03110          CALL    SCHTCO         ;CLEAR SCREE
N
47F6 21114E     03120          LD      HL,MSG19       ;DISPLAY BUF
FER FULL
47F9 CDA728     03130          CALL    28A7H          ;MESSAGE
47FC 18A0       03140          JR      STORE1         ;TO STORE ME
SSAGE ROUTINE
47FE CD2B00     03150 CM:      CALL    002BH          ;SCAN KEYS
4801 B7         03160          OR      A
4802 28FA       03170          JR      Z,CM           ;IF NO ENTRY
, CONTINUE
4804 FE60       03180          CP      60H            ;IF SHIFT @,
 TO
4806 2893       03190          JR      Z,STORE        ;MESSAGE MEN
U
4808 FE1B       03200          CP      1BH            ;IF SHIFT UP
 ARROW, TO
480A CA6347     03210          JP      Z,MON          ;MONITOR MEN
U
480D FE08       03220          CP      08H            ;IF BACKSPAC
E, DELETE LAST
480F 2002       03230          JR      NZ,CFIC
4811 180D       03240          JR      FHL
4813 FE0D       03250 CFIC:    CP      0DH            ;IF ENTER, P
LACE IN BUFFER
4815 2806       03260          JR      Z,OKFB
4817 FE20       03270          CP      20H            ;IGNORE OTHE
R CNTRL CHARS
4819 3F         03280          CCF
481A D2FE47     03290          JP      NC,CM
481D 77         03300 OKFB:    LD      (HL),A         ;PUT ENTRY I
N BUFFER
481E 1811       03310          JR      KHL            ;CHECK POSIT
ION IN BUFFER
4820 2B         03320 FHL:     DEC     HL             ;ADJUST POIN
TER
4821 2B         03330          DEC     HL
4822 E5         03340          PUSH    HL             ;SAVE POINTE
R
4823 2A2040     03350          LD      HL,(4020H)     ;GET CURSOR
POSITION
4826 AF         03360          XOR     A              ;CLEAR ACC
4827 11003C     03370          LD      DE,3C00H       ;SEE IF CURS
OR AT BEGIN-
482A ED52       03380          SBC     HL,DE          ;NING OF SCR
EEN
482C E1         03390          POP     HL             ;RETRIEVE BU
FFER POINTER
482D 2891       03400          JR      Z,STO          ;IF AT BEGIN
NING, CONTINUE
482F 3E08       03410          LD      A,08H          ;LOAD BACKSP
ACE INTO ACC
4831 CD3300     03420 KHL:     CALL    0033H          ;DISPLAY ACC
 CHARACTER
4834 23         03430          INC     HL             ;INC BUFFER
POINTER
4835 3600       03440          LD      (HL),00H       ;ZERO BUFFER
 BYTE
4837 F5         03450          PUSH    AF             ;SAVE CHARAC
TER
4838 E5         03460          PUSH    HL             ;SAVE POINTE
R
4839 110F47     03470          LD      DE,EMTERM      ;SEE IF AT E
ND OF
483C EB         03480          EX      DE,HL          ;BUFFER
483D AF         03490          XOR     A              ;CLEAR ACC
483E ED52       03500          SBC     HL,DE          ;SUBTRACT PO
INTER FROM END
4840 28AF       03510          JR      Z,BFM          ;OF BUFFER
4842 E1         03520          POP     HL             ;RETRIEVE PO
INTER
```

Program continues

✓ *See List of Advertisers on page 386*

Program continued

```
4843 F1         03530         POP     AF              ;RETRIEVE CH
ARACTER
4844 18B8       03540         JR      CM              ;CONTINUE EN
TERING
4846 CD5A4B     03550 TRA:    CALL    SCHTCO          ;CLEAR SCREE
N
4849 210043     03560         LD      HL,EMTERM-40FH  ;POINT TO BU
FFER START
484C 7E         03570 CTT:    LD      A,(HL)          ;GET BUFFER
BYTE
484D FE00       03580         CP      00H             ;IF ZERO, FI
NISHED
484F 2815       03590         JR      Z,RUART
4851 CD3300     03600         CALL    0033H           ;DISPLAY BUF
FER BYTE
4854 CDAD4B     03610         CALL    STTC            ;TRANSMIT BU
FFER BYTE
4857 CD1E4E     03620         CALL    STRMC           ;WAIT FOR EC
HO
485A 23         03630         INC     HL              ;INC POINTER

485B E5         03640         PUSH    HL              ;SAVE POINTE
R
485C 110F47     03650         LD      DE,EMTERM       ;SEE IF AT E
ND OF BUFFER
485F EB         03660         EX      DE,HL
4860 AF         03670         XOR     A               ;CLEAR ACC
4861 ED52       03680         SBC     HL,DE           ;SUBTRACT PO
INTER FROM END
4863 E1         03690         POP     HL              ;OF BUFFER
4864 20E6       03700         JR      NZ,CTT          ;IF NOT AT E
ND, CONTINUE
4866 CD9B4B     03710 RUART:  CALL    SRAPU           ;RESET AND P
ROGRAM UART
4869 CD5A4B     03720         CALL    SCHTCO          ;CLEAR SCREE
N
486C C3B249     03730         JP      TTU             ;BACK TO TER
MINAL MODE
486F 210043     03740 ERA:    LD      HL,EMTERM-40FH  ;POINT TO BU
FFER START
4872 3600       03750 KEB:    LD      (HL),00H        ;ZERO BUFFER
 BYTE
4874 23         03760         INC     HL              ;INC POINTER

4875 E5         03770         PUSH    HL              ;SEE IF ALL
THRU BUFFER
4876 110F47     03780         LD      DE,EMTERM
4879 EB         03790         EX      DE,HL
487A AF         03800         XOR     A               ;CLEAR ACC
487B ED52       03810         SBC     HL,DE           ;SUBTRACT PO
INTER FROM
487D E1         03820         POP     HL              ;END OF BUFF
ER
487E 20F2       03830         JR      NZ,KEB          ;KEEP ZEROIN
G BUFFER
4880 C39B47     03840         JP      STORE           ;TO STORE RO
UTINE
4883 CD5A4B     03850 REC:    CALL    SCHTCO          ;CLEAR SCREE
N
4886 CD2B00     03860 WFAS:   CALL    002BH           ;SCAN KEYS
4889 B7         03870         OR      A
488A 2807       03880         JR      Z,CSTRC         ;IF NO ENTRY
, REC CHAR?
488C FE1B       03890         CP      1BH             ;IF SHIFT UP
 ARROW, TO
488E 2003       03900         JR      NZ,CSTRC        ;MONITOR MEN
U
4890 C36347     03910         JP      MON
4893 CDD44D     03920 CSTRC:  CALL    STRBC           ;GET BASIC C
HARACTER
4896 FE41       03930         CP      'A'             ;WAIT FOR TH
REE A'S
4898 20EC       03940         JR      NZ,WFAS         ;IN SEQUENCE

489A CDD44D     03950         CALL    STRBC
489D FE41       03960         CP      'A'
489F 20E5       03970         JR      NZ,WFAS
48A1 CDD44D     03980         CALL    STRBC
48A4 FE41       03990         CP      'A'
48A6 20DE       04000         JR      NZ,WFAS
48A8 21BC4D     04010         LD      HL,MSG17        ;DISPLAY REC
EPTION MESSAGE
48AB CDA728     04020         CALL    28A7H
48AE 21E942     04030         LD      HL,42E9H        ;POINT TO ST
```

Program continues

Program continued

```
ART OF BASIC
48B1 CDD44D    04040 CRP:    CALL    STRBC        ;GET BASIC C
HARACTER
48B4 FE00      04050         CP      00H          ;LOOK FOR FI
RST OF THREE
48B6 2022      04060         JR      NZ,CFMS      ;ZEROS IN SE
QUENCE
48B8 77        04070         LD      (HL),A       ;SAVE BASIC
CHARACTER
48B9 23        04080         INC     HL           ;POINT TO NE
XT BUFFER BYTE
48BA CD2C02    04090         CALL    22CH         ;BLINK ASTER
ISK
48BD CDD44D    04100         CALL    STRBC        ;GET CHARACT
ER
48C0 FE00      04110         CP      00H          ;LOOK FOR SE
COND ZERO
48C2 2016      04120         JR      NZ,CFMS      ;IN SEQUENCE

48C4 77        04130         LD      (HL),A       ;SAVE BASIC
CHARACTER
48C5 23        04140         INC     HL           ;POINT TO NE
XT BUFFER BYTE
48C6 CDD44D    04150         CALL    STRBC        ;GET BASIC C
HARACTER
48C9 FE00      04160         CP      00H          ;LOOK FOR TH
IRD ZERO
48CB 200D      04170         JR      NZ,CFMS      ;IN SEQUENCE

48CD 77        04180         LD      (HL),A       ;SAVE BASIC
CHARACTER
48CE 23        04190         INC     HL           ;INC POINTER

48CF 7D        04200         LD      A,L          ;SAVE NEW BA
SIC ENDING
48D0 32F940    04210         LD      (40F9H),A    ;ADDRESS
48D3 7C        04220         LD      A,H
48D4 32FA40    04230         LD      (40FAH),A
48D7 C36347    04240         JP      MON          ;BACK TO MON
ITOR MENU
48DA 77        04250 CFMS:   LD      (HL),A       ;PUT BASIC C
HARACTER
48DB 23        04260         INC     HL           ;INTO BUFFER

48DC CD2B00    04270         CALL    002BH        ;SCAN KEYS
48DF B7        04280         OR      A
48E0 28CF      04290         JR      Z,CRP        ;IF NO ENTRY
, CONTINUE
48E2 FE1B      04300         CP      1BH          ;IF SHIFT UP
 ARROW, TO
48E4 CA6347    04310         JP      Z,MON        ;MONITOR MEN
U
48E7 18C8      04320         JR      CRP          ;CONTINUE RE
CEIVING PROG
48E9 CD5A4B    04330 XMIT:   CALL    SCHTCO       ;CLEAR SCREE
N
48EC 21674D    04340         LD      HL,MSG12     ;DISPLAY TRA
NSMIT MENU
48EF CDA728    04350         CALL    28A7H
48F2 CD2B00    04360 WFTLE:  CALL    002BH        ;SCAN KEYS
48F5 B7        04370         OR      A
48F6 28FA      04380         JR      Z,WFTLE      ;WAIT FOR EN
TRY
48F8 FE4C      04390         CP      'L'          ;IF L, TO LO
AD CASSETTE
48FA 280E      04400         JR      Z,LOAD       ;ROUTINE
48FC FE54      04410         CP      'T'          ;IF T, TO TR
ANSMIT ROUTINE
48FE 2863      04420         JR      Z,TRANS
4900 FE1B      04430         CP      1BH          ;IF SHIFT UP
 ARROW, TO
4902 CA6347    04440         JP      Z,MON        ;MONITOR MEN
U
4905 CD944B    04450         CALL    SERROR       ;DISPLAY ERR
OR MESSAGE
4908 18E8      04460         JR      WFTLE        ;WAIT FOR EN
TRY
490A 218C4D    04470 LOAD:   LD      HL,MSG13     ;DISPLAY CAS
SETTE MESSAGE
490D CDA728    04480         CALL    28A7H
4910 CD2B00    04490 WFLK:   CALL    002BH        ;SCAN KEYS
4913 B7        04500         OR      A
4914 28FA      04510         JR      Z,WFLK       ;WAIT FOR EN
```

Program continues

✓See List of Advertisers on page 386

Program continued

```
TRY
4916 FE4C        04520        CP      'L'          ;IF L, TO LO
AD ROUTINE
4918 280A        04530        JR      Z,LT
491A FE1B        04540        CP      1BH          ;IF SHIFT UP
 ARROW, TO
491C CA6347      04550        JP      Z,MON        ;MONITOR MEN
U
491F CD944B      04560        CALL    SERROR       ;DISPLAY ERR
OR MESSAGE
4922 18EC        04570        JR      WFLK         ;WAIT FOR EN
TRY
4924 CD9302      04580 LT:    CALL    293H         ;TURN ON CAS
SETTE
4927 0604        04590        LD      B,04H        ;IGNORE FIRS
T FOUR BYTES
4929 CD3502      04600 GTH:   CALL    235H         ;GET TAPE CH
ARACTER
492C 10FB        04610        DJNZ    GTH          ;COMPLETE LO
OP
492E 21E942      04620        LD      HL,42E9H     ;POINT TO ST
ART OF BASIC
4931 CD2B00      04630 GNC:   CALL    002BH        ;SCAN KEYS
4934 B7          04640        OR      A
4935 280A        04650        JR      Z,CON        ;IF NO ENTRY
, CONTINUE
4937 FE1B        04660        CP      1BH          ;IF SHIFT UP
 ARROW, TO
4939 2006        04670        JR      NZ,CON       ;MONITOR MEN
U
493B CDF801      04680        CALL    1F8H         ;TURN OFF CA
SSETTE
493E C36347      04690        JP      MON          ;TO MONITOR
MENU
4941 CDA74B      04700 CON:   CALL    STGC         ;GET TAPE CH
ARACTER
4944 A7          04710        AND     A            ;1ST 0 IN SE
QUENCE?
4945 20EA        04720        JR      NZ,GNC       ;GET NEXT CH
ARACTER
4947 CD2C02      04730        CALL    22CH         ;ZERO, BLINK
 ASTERISK
494A CDA74B      04740        CALL    STGC         ;GET TAPE CH
ARACTER
494D A7          04750        AND     A            ;2ND 0 IN SE
QUENCE?
494E 20E1        04760        JR      NZ,GNC       ;GET NEXT CH
ARACTER
4950 CDA74B      04770        CALL    STGC         ;GET TAPE CH
ARACTER
4953 A7          04780        AND     A            ;3RD 0 IN SE
QUENCE?
4954 20DB        04790        JR      NZ,GNC       ;GET NEXT CH
ARACTER
4956 7D          04800        LD      A,L          ;SAVE NEW EN
DING ADDRESS
4957 32F940      04810        LD      (40F9H),A
495A 7C          04820        LD      A,H
495B 32FA40      04830        LD      (40FAH),A
495E CDF801      04840        CALL    1F8H         ;TURN OFF CA
SSETTE
4961 1886        04850        JR      XMIT         ;TO TRANSMIT
 MINI MENU
4963 21AB4D      04860 TRANS: LD      HL,MSG14     ;DISPLAY TRA
NSMIT MESSAGE
4966 CDA728      04870        CALL    28A7H
4969 21E942      04880        LD      HL,42E9H     ;POINT TO ST
ART OF BASIC
496C 0603        04890        LD      B,03H        ;SEND THREE
A'S FOR SYNC
496E 3E41        04900 KSA:   LD      A,'A'
4970 CDAD4B      04910        CALL    STTC         ;TRANSMIT CH
ARACTER
4973 10F9        04920        DJNZ    KSA          ;COMPLETE LO
OP
4975 CD2B00      04930 CFMR:  CALL    002BH        ;SCAN KEYS
4978 B7          04940        OR      A
4979 2805        04950        JR      Z,TRCH       ;NO ENTRY, T
RANSMIT CHAR
497B FE1B        04960        CP      1BH          ;IF SHIFT UP
 ARROW, TO
497D C36347      04970        JP      MON          ;MONITOR MEN
U
4980 7E          04980 TRCH:  LD      A,(HL)       ;GET BASIC C
```

Program continues

The Captain 80 Book of

BASIC ADVENTURES

• Over a dozen famous adventures, listed in large easy to read type, ready to be keyed in. Adventures by Boner, Kepner, Powers, Micklus, Forsythe, Greg Hassett and others.

• Includes *Temple of the Sun, Dog Star, Lost Ship, Spider Mountain, Lost Dutchman's Gold, Thunder Road, Sorcerer's Castle, Deadly Dungeon, Atlantean Oddysey* and others. These adventures would cost over $200 is purchased individually!

• Includes a unique *Adventure Generator* program - not available anywhere. YES, this program will actually write another BASIC adventure program, which you may then run. Not even the author of the generator program knows the outcome! *This program alone is worth the price of the entire book!*

• Includes chapters on what an adventure is, how to play adventures, how to write adventures, how to sell your adventure, ten adventure program ideas, and more!

• Study the techniques and methods used by the masters of adventure program writing. All adventure programs listed in this book are in their original form and in full length. Although specifically written for the TRS-80 Model I and III, these programs are adaptable to other computers using Microsoft's BASIC.

• Available at your dealer approximately November 15th, 1981. The suggested retail price of this book is $19.95.

• The BASIC Adventure Book was compiled and edited by Bob Liddil, of The Programmers Guild. It measures 8½ by 11 inches and will contain approximately 250 pages.

• To reserve your copy of this unique book, order NOW for only $14.95 and save. This special pre-publication offer is good only with paid orders until Nov 1st, 1981.

• *ORDER YOURS TODAY!*

Dealer discounts available - please inquire

TRS-80® is a registered trademark of the Tandy Corp.

Pre-Publication SPECIAL $14.95

Send To: 80-U.S. Journal
3838 South Warner Street
Tacoma, Washington 98409 ✓33

Please reserve ____ Copy (Copies) of the Basic Adventure Book at the pre-publication price of only $14.95. Payment must be enclosed with this order. Offer good in the United States only.

Name________________________

Address ________________________

City__________ State_____ Zip ________

Visa/MC________________________

Exp. Date_____ Signature ____________________

(206) 475-2219 Phone orders accepted for Visa/MC only.

ORDER # 8M

✓See List of Advertisers on page 386

Program continued

```
HARACTER
4981 FE00      04990          CP      00H            ;1ST 0 IN SE
QUENCE?
4983 201A      05000          JR      NZ,CSTTC       ;IF NO 0, TR
ANSMIT CHAR
4985 CDAD4B    05010          CALL    STTC           ;TRANSMIT ZE
RO
4988 23        05020          INC     HL             ;INC POINTER
4989 7E        05030          LD      A,(HL)         ;GET NEXT BA
SIC CHARACTER
498A FE00      05040          CP      00H            ;2ND 0 IN SE
QUENCE?
498C 2011      05050          JR      NZ,CSTTC       ;IF NO 0, TR
ANSMIT CHAR
498E CDAD4B    05060          CALL    STTC           ;TRANSMIT SE
COND ZERO IN
4991 23        05070          INC     HL             ;SEQUENCE &
INC POINTER
4992 7E        05080          LD      A,(HL)         ;GET NEXT BA
SIC CHARACTER
4993 FE00      05090          CP      00H            ;LOOK FOR TH
IRD ZERO IN
4995 2008      05100          JR      NZ,CSTTC       ;SEQUENCE, T
RANS CHAR
4997 CDAD4B    05110          CALL    STTC           ;TRANSMIT TH
IRD ZERO IN
499A C3E948    05120          JP      XMIT           ;SEQUENCE &
TO MINI MENU
499D 00        05130          NOP
499E 00        05140          NOP
499F CDAD4B    05150 CSTTC:   CALL    STTC           ;TRANSMIT CH
AR ROUTINE
49A2 23        05160          INC     HL             ;INC POINTER

49A3 18D0      05170          JR      CFMR           ;CHECK FOR M
ONITOR RETURN
49A5 CD5A4B    05180 TERM:    CALL    SCHTCO         ;CLEAR SCREE
N
49A8 AF        05190          XOR     A              ;CLEAR ACC
49A9 32D14D    05200          LD      (ERROR),A      ;ZERO ERROR,
 PRINT, CONTRL
49AC 32D24D    05210          LD      (PRINT),A      ;RAM LOCATIO
NS
49AF 32D34D    05220          LD      (CONTRL),A
49B2 CD2B00    05230 TTU:     CALL    002BH          ;SCAN KEYS
49B5 B7        05240          OR      A
49B6 284A      05250          JR      Z,RFU          ;CHECK FOR R
ECEIVED CHAR
49B8 FE1A      05260          CP      1AH            ;CHECK FOR C
ONTROL CHAR
49BA 200E      05270          JR      NZ,CFSAT       ;IF NONE, CH
ECK FOR SHIFT@
49BC 3EFF      05280          LD      A,0FFH         ;LOAD CONTRL
 WITH 0FFH TO
49BE 32D34D    05290          LD      (CONTRL),A     ;NOTE CONTRO
L CHAR
49C1 CD2B00    05300 WFCC:    CALL    002BH          ;SCAN KEYS
49C4 B7        05310          OR      A
49C5 28FA      05320          JR      Z,WFCC         ;WAIT FOR EN
TRY
49C7 C3FF49    05330          JP      NCD            ;TRANSMIT CH
ARACTER
49CA F5        05340 CFSAT:   PUSH    AF             ;SAVE CHAR
49CB FE60      05350          CP      60H            ;IF SHIFT@,
SWITCH ERROR
49CD 200A      05360          JR      NZ,CFSLA       ;DISPLAY STA
TUS . OTHER-
49CF 3AD14D    05370          LD      A,(ERROR)      ;WISE CHECK
FOR SHIFT<-
49D2 2F        05380          CPL
49D3 32D14D    05390          LD      (ERROR),A
49D6 F1        05400          POP     AF             ;RETRIEVE CH
ARACTER
49D7 1829      05410          JR      RFU            ;CHECK FOR R
ECEIVED CHAR
49D9 FE18      05420 CFSLA:   CP      18H            ;CHECK FOR S
HIFT<-
49DB 200A      05430          JR      NZ,GAB         ;IF NOT, GET
 CHAR BACK
49DD 3AD24D    05440          LD      A,(PRINT)      ;CHANGE PRIN
T STATUS
49E0 2F        05450          CPL
```

Program continues

Program continued

```
49E1 32D24D     05460            LD        (PRINT),A
49E4 F1         05470            POP       AF               ;RETRIEVE CH
ARACTER
49E5 181B       05480            JR        RFU              ;CHECK FOR R
ECEIVED CHAR
49E7 F1         05490 GAB:       POP       AF               ;RETRIEVE CH
ARACTER
49E8 FE1B       05500 CFRTM:     CP        1BH              ;IF SHIFT UP
 ARROW, TO
49EA CA6347     05510            JP        Z,MON            ;MONITOR MEN
U
49ED FE01       05520            CP        01H              ;IF BREAK, T
O 0'S ROUTINE
49EF 283F       05530            JR        Z,BREAK
49F1 FE19       05540            CP        19H              ;IF SHIFT->,
 TO TRANSMIT
49F3 CA4648     05550            JP        Z,TRA            ;MESSAGE ROU
TINE
49F6 FE1F       05560            CP        1FH              ;IF CLEAR, T
O CLEAR SCREEN
49F8 2005       05570            JR        NZ,NCD           ;ROUTINE
49FA CD5A4B     05580            CALL      SCHTCO           ;CLEAR SCREE
N
49FD 1803       05590            JR        RFU              ;CHECK FOR R
ECEIVED CHAR
49FF CDAD4B     05600 NCD:       CALL      STTC             ;TRANSMIT CH
ARACTER
4A02 CDB84B     05610 RFU:       CALL      STRC             ;CHECK FOR R
ECEIVED CHAR
4A05 B7         05620            OR        A
4A06 28AA       05630            JR        Z,TTU            ;BACK TO SCA
N KEYS
4A08 F5         05640            PUSH      AF               ;SAVE RECEIV
ED CHARACTER
4A09 B9         05650            CP        C                ;CHAR REC=CH
AR SENT?
4A0A 2802       05660            JR        Z,SICC           ;IF SO, SEE
IF CNTRL CHAR
4A0C 1811       05670            JR        TFP              ;CHECK FOR P
RINT ENABLE
4A0E FE20       05680 SICC:      CP        20H              ;IF CONTROL
CHARACTER, WAS
4A10 300D       05690            JR        NC,TFP           ;IT ECHOED?
4A12 3AD34D     05700            LD        A,(CONTRL)
4A15 B7         05710            OR        A
4A16 2807       05720            JR        Z,TFP            ;IF SO, IGNO
RE IT
4A18 F1         05730            POP       AF
4A19 AF         05740            XOR       A
4A1A 32D34D     05750            LD        (CONTRL),A       ;CLEAR CONTR
L LOCATION
4A1D 1893       05760            JR        TTU              ;BACK TO SCA
N KEYS
4A1F 3AD24D     05770 TFP:       LD        A,(PRINT)        ;IF PRINTER
ENABLED, PRINT
4A22 B7         05780            OR        A                ;INCOMING CH
ARACTER
4A23 2805       05790            JR        Z,TDO
4A25 F1         05800            POP       AF
4A26 32E837     05810            LD        (37E8H),A
4A29 F5         05820            PUSH      AF
4A2A F1         05830 TDO:       POP       AF
4A2B CD3300     05840            CALL      0033H            ;DISPLAY INC
OMING CHAR
4A2E 1882       05850            JR        TTU              ;BACK TO SCA
N KEYS
4A30 3ACF4D     05860 BREAK:     LD        A,(HUCR)         ;ZERO SERIAL
 OUTPUT PORT
4A33 E6FB       05870            AND       0FBH             ;FOR A SHORT
 TIME TO
4A35 D3EA       05880            OUT       (0EAH),A         ;GENERATE BR
EAK SIGNAL
4A37 21FFFF     05890            LD        HL,0FFFFH        ;DELAY ROUTI
NE
4A3A 2B         05900 GDLY:      DEC       HL               ;DECREMENT H
L UNTIL
4A3B 7C         05910            LD        A,H              ;ZERO RESULT

4A3C B5         05920            OR        L
4A3D 20FB       05930            JR        NZ,GDLY
4A3F 3ACF4D     05940            LD        A,(HUCR)         ;ENABLE SERI
AL OUTPUT PORT
4A42 E6F8       05950            AND       0F8H
```

Program continues

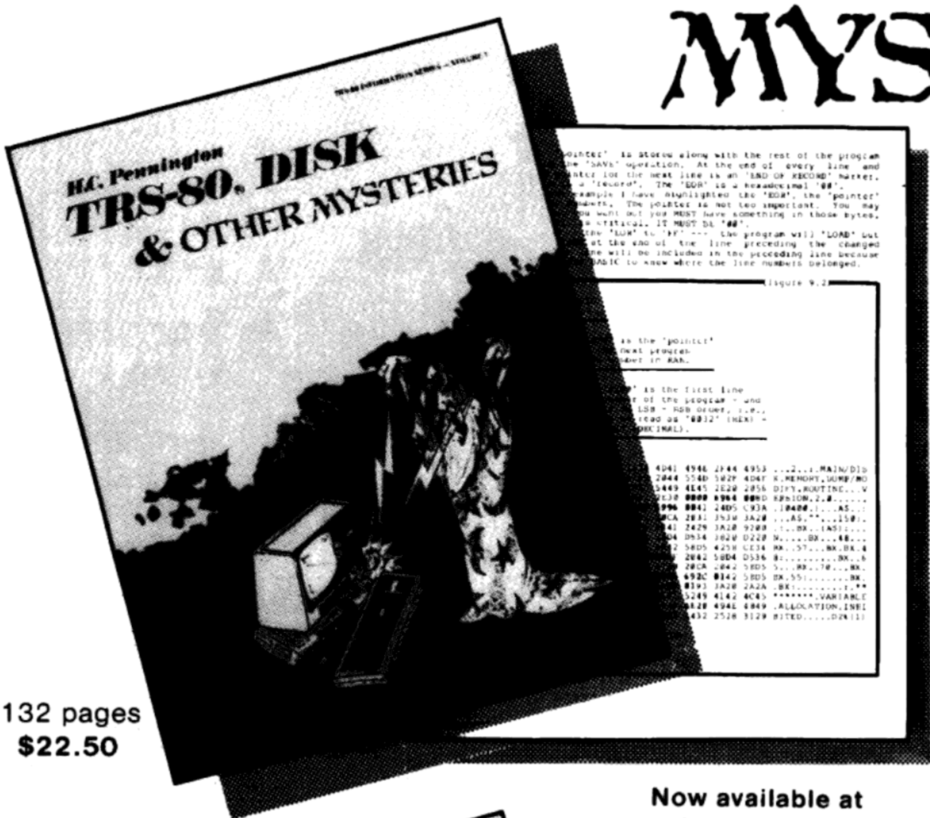
Program continued

```
4A44 F605      05960          OR     05H
4A46 D3EA      05970          OUT    (0EAH),A
4A48 18B8      05980          JR     RFU           ;CHECK FOR R
EC CHAR
4A4A CD5A4B    05990 VIEW:    CALL   SCHTCO        ;CLEAR SCREE
N
4A4D 21404C    06000          LD     HL,MSG1       ;DISPLAY MEN
U
4A50 CDA728    06010          CALL   28A7H
4A53 21CF4D    06020          LD     HL,HUCR       ;READ UART C
ONTROL LOC
4A56 CB5E      06030          BIT    3,(HL)        ;CHECK PARIT
Y & DISPLAY
4A58 2014      06040          JR     NZ,DISMSG     ;IF ODD, EVE
N, OR NONE
4A5A CB7E      06050          BIT    7,(HL)
4A5C 2008      06060          JR     NZ,EMSG
4A5E 21554C    06070 OMSG:    LD     HL,MSG3       ;"ODD" DISPL
AY
4A61 CDA728    06080          CALL   28A7H
4A64 180E      06090          JR     CCMSG1
4A66 21594C    06100 EMSG:    LD     HL,MSG4       ;"EVEN" DISP
LAY
4A69 CDA728    06110          CALL   28A7H
4A6C 1806      06120          JR     CCMSG1
4A6E 214C4C    06130 DISMSG:  LD     HL,MSG2       ;"NONE" DISP
LAY
4A71 CDA728    06140          CALL   28A7H
4A74 CD6A4B    06150 CCMSG1:  CALL   SCMSG         ;CHANGE DESI
RED?
4A77 B7        06160          OR     A
4A78 2838      06170          JR     Z,WLMSG       ;IF NOT, DIS
PLAY WD LENGTH
4A7A FEAA      06180          CP     0AAH          ;IF AA, TO M
ONITOR MENU
4A7C CA6347    06190          JP     Z,MON
4A7F 216E4C    06200          LD     HL,MSG6       ;DISPLAY PAR
ITY CHOICES
4A82 CDA728    06210          CALL   28A7H
4A85 CD2B00    06220 NPS:     CALL   002BH         ;WAIT FOR O,
 E, OR N
4A88 B7        06230          OR     A             ;OR SHIFT UP
 ARROW
4A89 28FA      06240          JR     Z,NPS
4A8B 21CF4D    06250          LD     HL,HUCR
4A8E FE4F      06260          CP     'O'           ;IF "O", SET
 ODD
4A90 2812      06270          JR     Z,OIS
4A92 FE45      06280          CP     'E'           ;IF "E", SET
 EVEN
4A94 2814      06290          JR     Z,EIC
4A96 FE4E      06300          CP     'N'           ;IF "N", SET
 NO PARITY
4A98 2816      06310          JR     Z,DIC
4A9A FE1B      06320          CP     1BH           ;IF SHIFT UP
 ARROW, TO
4A9C CA6347    06330          JP     Z,MON         ;MONITOR MEN
U
4A9F CD944B    06340          CALL   SERROR        ;DISPLAY ERR
OR MESSAGE
4AA2 18E1      06350          JR     NPS           ;WAIT FOR AN
OTHER ENTRY
4AA4 CBBE      06360 OIS:     RES    7,(HL)        ;SET UP ODD
PARITY
4AA6 CB9E      06370          RES    3,(HL)
4AA8 1808      06380          JR     WLMSG         ;TO WL DISPL
AY
4AAA CBFE      06390 EIC:     SET    7,(HL)        ;SET UP EVEN
 PARITY
4AAC CB9E      06400          RES    3,(HL)
4AAE 1802      06410          JR     WLMSG         ;TO WL DISPL
AY
4AB0 CBDE      06420 DIC:     SET    3,(HL)        ;DISABLE PAR
ITY
4AB2 219B4C    06430 WLMSG:   LD     HL,MSG8       ;DISPLAY WOR
D LENGTH
4AB5 CDA728    06440          CALL   28A7H
4AB8 3ACF4D    06450          LD     A,(HUCR)      ;READ UART C
ONTROL LOC
4ABB E660      06460          AND    60H           ;ISOLATE WL
BITS
4ABD FE00      06470          CP     00H           ;IF 5 BITS,
DISPLAY
```

Program continues

'.. this is not only a worthwhile book but a great book. My advice is to get it and USE it!' - **William Barden Jr.**

OTHER MYSTERIES



132 pages
$22.50

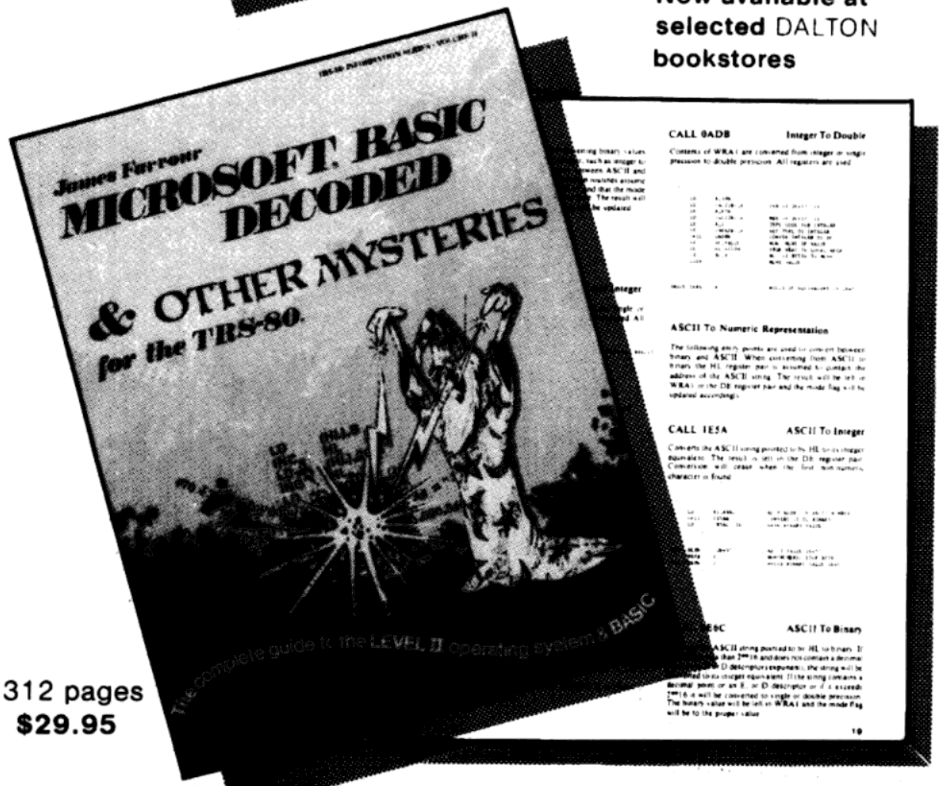
TRS-80 Disk and Other Mysteries is the definitive **fixit** book for disk users. Writen by Harvard Pennington it has more than 130 pages of easy to read, entertaining and immensely useful information - find out how to recover disk files, the layout of information on the disk, memory maps, problem solutions . . . the list goes on!

Many readers have saved days of work by recovering files that had been unreadable. Now in its fourth updated printing, **TRS-80 Disk** and Other Mysteries has been getting rave reviews in several magazines. Yours for only **$22.50** (plus $1.50 shipping, CA residents please add $1.35 sales tax).

Microsoft BASIC Decoded and Other Mysteries is the complete guide to your Level II ROMs. With over seven thousand lines of comments and 6 additional chapters packed with information, this is easily the biggest, and best, book about the Level II ROMs available.

Exploit the full power of Microsoft BASIC, with the aid of more than 300 pages of tested examples, understandable explanations and detailed comments. Now available in a revised second printing, only **$29.95** (plus $2.00 shipping, CA residents add $1.80 sales tax).

Now available at selected DALTON **bookstores**



312 pages
$29.95

Get them at your local IJG dealer!

'It has twelve thousand one-liners in it, and every one is great!'
- **Dennis Kitsz.**

Phone orders (714) 946-5805

TRS-80 is a trademark of Tandy, Microsoft is trademark of Microsoft.

IJG Computer Services, ✓37
1260 West Foothill Blvd.,
Upland, CA 91786

Please send me a copy of **TRS-80 Disk** and Other Mysteries for $22.50 plus $1.50 shipping. ☐

Please send me a copy of **Microsoft BASIC Decoded** and Other Mysteries for $29.95 plus $2.00 shipping. ☐

Please send me a copy of **The Custom TRS-80** and Other Mysteries for $29.95 plus $2.00 shipping. ☐

OK, send me all three **IJG** books for $82.40 plus $5.00 shipping. ☐

Overseas add $8.00 per book airmail.

Name ....

Address ....

City ....

State .... Zip ....

Charge my:

☐ MasterCard Interbank Code ....

☐ Visa Expiration Date ....

Card No. ....

☐ Check enclosed for ....

☐ Ship COD ($2.00 extra)

Signature ....

✓See List of Advertisers on page 386

Program continued

```
4ABF 2004      06480        JR     NZ,CF7
4AC1 3E35      06490        LD     A,'5'
4AC3 1812      06500        JR     DIS
4AC5 FE20      06510 CF7:   CP     20H           ;IF 7 BITS,
DISPLAY
4AC7 2004      06520        JR     NZ,CF6
4AC9 3E37      06530        LD     A,'7'
4ACB 180A      06540        JR     DIS
4ACD FE40      06550 CF6:   CP     40H           ;IF 6 BITS,
DISPLAY
4ACF 2004      06560        JR     NZ,MB8
4AD1 3E36      06570        LD     A,'6'
4AD3 1802      06580        JR     DIS
4AD5 3E38      06590 MB8:   LD     A,'8'         ;DISPLAY 8 B
IT WL
4AD7 CD3300    06600 DIS:   CALL   0033H
4ADA CD6A4B    06610        CALL   SCMSG         ;CHANGE WORD
 LENGTH?
4ADD B7        06620        OR     A
4ADE 2844      06630        JR     Z,SBMSG       ;IF NOT SHOW
 STOP BITS
4AE0 FEAA      06640        CP     0AAH          ;IF AA, TO M
ONITOR MENU
4AE2 CA6347    06650        JP     Z,MON
4AE5 21AB4C    06660        LD     HL,MSG9       ;DISPLAY WOR
D LENGTH
4AE8 CDA728    06670        CALL   28A7H         ;CHOICES
4AEB CD2B00    06680 WFWL:  CALL   002BH         ;WAIT FOR SE
LECTION OR
4AEE 21CF4D    06690        LD     HL,HUCR       ;RETURN TO M
ONITOR MENU
4AF1 B7        06700        OR     A
4AF2 28F7      06710        JR     Z,WFWL
4AF4 FE35      06720        CP     '5'           ;IF "5", SET
 UP
4AF6 2828      06730        JR     Z,FIV
4AF8 FE36      06740        CP     '6'           ;IF "6", SET
 UP
4AFA 281E      06750        JR     Z,SIX
4AFC FE37      06760        CP     '7'           ;IF "7", SET
 UP
4AFE 2814      06770        JR     Z,SEV
4B00 FE38      06780        CP     '8'           ;IF "8", SET
 UP
4B02 280A      06790        JR     Z,EIG
4B04 FE1B      06800        CP     1BH           ;IF SHIFT UP
 ARROW, TO
4B06 CA6347    06810        JP     Z,MON         ;MONITOR MEN
U
4B09 CD944B    06820        CALL   SERROR        ;DISPLAY ERR
OR MESSAGE
4B0C 18DD      06830        JR     WFWL          ;WAIT FOR AN
OTHER ENTRY
4B0E CBF6      06840 EIG:   SET    6,(HL)        ;SET UP 8 BI
T WL
4B10 CBEE      06850        SET    5,(HL)
4B12 1810      06860        JR     SBMSG
4B14 CBB6      06870 SEV:   RES    6,(HL)        ;SET UP 7 BI
T WL
4B16 CBEE      06880        SET    5,(HL)
4B18 180A      06890        JR     SBMSG
4B1A CBF6      06900 SIX:   SET    6,(HL)        ;SET UP 6 BI
T WL
4B1C CBAE      06910        RES    5,(HL)
4B1E 1804      06920        JR     SBMSG
4B20 CBB6      06930 FIV:   RES    6,(HL)        ;SET UP 5 BI
T WL
4B22 CBAE      06940        RES    5,(HL)
4B24 218B4C    06950 SBMSG: LD     HL,MSG7       ;DISPLAY STO
P BIT MESSAGE
4B27 CDA728    06960        CALL   28A7H
4B2A 21CF4D    06970        LD     HL,HUCR       ;READ UART C
ONTROL LOC
4B2D CB66      06980        BIT    4,(HL)        ;ISOLATE STO
P BIT BIT
4B2F 2007      06990        JR     NZ,TSB
4B31 3E31      07000        LD     A,'1'         ;IF "0", DIS
PLAY 1
4B33 CD3300    07010        CALL   0033H
4B36 1805      07020        JR     CCMSG2
4B38 3E32      07030 TSB:   LD     A,'2'         ;IF "1", DIS
PLAY 2
4B3A CD3300    07040        CALL   0033H
4B3D CD6A4B    07050 CCMSG2: CALL  SCMSG         ;CHANGE DESI
RED?
```

Program continues

Program continued

```
4B40 B7                 OR     A
4B41 CA6347             JP     Z,MON          ;IF 00 OR AA
, TO MON-
4B44 FEAA               CP     0AAH           ;ITOR MENU
4B46 CA6347             JP     Z,MON
4B49 21CF4D             LD     HL,HUCR        ;CHANGE # OF
 STOP BITS
4B4C CB66               BIT    4,(HL)         ;CHANGE BIT
 4 OF UART
4B4E 2805               JR     Z,SB4          ;CONTROL LOC
ATION
4B50 CBA6               RES    4,(HL)
4B52 C36347             JP     MON            ;BACK TO MON
ITOR MENU
4B55 CBE6       SB4:    SET    4,(HL)
4B57 C36347             JP     MON
4B5A 3E1C       SCHTCO: LD     A,1CH          ;CLEAR SCREE
N, HOME CURSOR
4B5C CD3300             CALL   0033H          ;SUBROUTINE
4B5F 3E1F               LD     A,1FH
4B61 CD3300             CALL   0033H
4B64 3E0E               LD     A,0EH
4B66 CD3300             CALL   0033H
4B69 C9                 RET
4B6A 215E4C     SCMSG:  LD     HL,MSG5        ;DISPLAY CHA
NGE QUESTION
4B6D CDA728             CALL   28A7H
4B70 CD2B00     WFCA:   CALL   002BH          ;SCAN KEYS F
OR Y, N,
4B73 B7                 OR     A              ;ENTER, OR S
HIFT UP ARROW
4B74 28FA               JR     Z,WFCA
4B76 FE4E               CP     'N'            ;IF N, ZERO
ACC
4B78 2811               JR     Z,ZACC
4B7A FE0D               CP     0DH            ;IF ENTER, Z
ERO ACC
4B7C 280D               JR     Z,ZACC
4B7E FE59               CP     'Y'            ;IF Y, ACC =
 FF
4B80 280C               JR     Z,SACC
4B82 FE1B               CP     1BH            ;IF SHIFT UP
 ARROW, ACC=AA
4B84 280B               JR     Z,RTM
4B86 CD944B             CALL   SERROR         ;DISPLAY ERR
OR MESSAGE
4B89 18E5               JR     WFCA           ;WAIT FOR EN
TRY
4B8B 3E00       ZACC:   LD     A,00H          ;00 ON RETUR
N=NO
4B8D C9                 RET
4B8E 3EFF       SACC:   LD     A,0FFH         ;FF ON RETUR
N=YES
4B90 C9                 RET
4B91 3EAA       RTM:    LD     A,0AAH         ;AA ON RETUR
N=RETURN TO
4B93 C9                 RET                   ;MONITOR MEN
U
4B94 21584D     SERROR: LD     HL,MSG11       ;SUBROUTINE
TO DISPLAY
4B97 CDA728             CALL   28A7H          ;ILLEGAL ENT
RY ON CRT
4B9A C9                 RET
4B9B D3E8       SRAPU:  OUT    (0E8H),A       ;SUBROUTINE
TO RESET
4B9D 3ACF4D             LD     A,(HUCR)       ;AND PROGRAM
 UART
4BA0 E6F8               AND    0F8H           ;FROM HOLDIN
G LOCATION
4BA2 F605               OR     05H
4BA4 D3EA               OUT    (0EAH),A
4BA6 C9                 RET
4BA7 CD3502     STGC:   CALL   235H           ;SUBROUTINE
TO GET CHAR
4BAA 77                 LD     (HL),A         ;FROM CASSET
TE TAPE
4BAB 23                 INC    HL             ;AND INC BUF
FER POINTER
4BAC C9                 RET
4BAD 4F         STTC:   LD     C,A            ;SUBROUTINE
TO TRANSMIT
4BAE DBEA       WFU:    IN     A,(0EAH)       ;CHARACTER I
N ACC
```

Program continues

Program continued

```
4BB0 CB77      07600          BIT     6,A             ;WAIT FOR EM
PTY TRANSMIT
4BB2 28FA      07610          JR      Z,WFU           ;BUFFER
4BB4 79        07620          LD      A,C
4BB5 D3EB      07630          OUT     (0EBH),A        ;TRANSMIT CH
ARACTER
4BB7 C9        07640          RET
4BB8 DBEA      07650 STRC:    IN      A,(0EAH)        ;SUBROUTINE
TO RECEIVE
4BBA 32D04D    07660          LD      (HUSR),A        ;CHARACTERS
FROM UART
4BBD CB7F      07670          BIT     7,A             ;TEST FOR RE
C CHAR READY
4BBF 2820      07680          JR      Z,RETWZ         ;RETURN IF N
OT
4BC1 DBEB      07690          IN      A,(0EBH)        ;GET CHARACT
ER
4BC3 F5        07700          PUSH    AF
4BC4 3AD14D    07710          LD      A,(ERROR)       ;SEE IF ERRO
R DISPLAY ON
4BC7 B7        07720          OR      A
4BC8 2007      07730          JR      NZ,DISDAT       ;IF NOT, DIS
PLAY CHARACTER
4BCA 3AD04D    07740          LD      A,(HUSR)        ;GET SAVED U
ART STATUS
4BCD E638      07750          AND     38H             ;ISOLATE ERR
OR BITS
4BCF 2012      07760          JR      NZ,ERR          ;IF NON-ZERO
, FIND ERROR
4BD1 F1        07770 DISDAT:  POP     AF              ;RETRIEVE CH
ARACTER
4BD2 E67F      07780          AND     7FH             ;STRIP 8TH B
IT
4BD4 FE60      07790          CP      60H             ;IF SMALLS,
CHANGE TO CAPS
4BD6 3F        07800          CCF
4BD7 D2DC4B    07810          JP      NC,CFLF
4BDA CBAF      07820          RES     5,A
4BDC FE0A      07830 CFLF:    CP      0AH             ;IF LINE FEE
D, IGNORE
4BDE 2801      07840          JR      Z,RETWZ         ;RETURN WITH
 ACC = 0
4BE0 C9        07850          RET
4BE1 AF        07860 RETWZ:   XOR     A               ;CLEAR ACC
4BE2 C9        07870          RET                     ;RETURN
4BE3 21D04D    07880 ERR:     LD      HL,HUSR         ;READ SAVED
ERROR STATUS
4BE6 CB6E      07890          BIT     5,(HL)          ;IF OVERRUN,
 DISPLAY "O"
4BE8 C4F74B    07900          CALL    NZ,OE
4BEB CB66      07910          BIT     4,(HL)          ;IF FRAMING,
 DISPLAY "F"
4BED C4FD4B    07920          CALL    NZ,FE
4BF0 CB5E      07930          BIT     3,(HL)          ;IF PARITY,
DISPLAY "P"
4BF2 C4014C    07940          CALL    NZ,PAR          ;TO DISPLAY
BAR
4BF5 180E      07950          JR      DISB
4BF7 3E4F      07960 OE:      LD      A,'O'           ;LOAD "O" CO
DE
4BF9 CD3300    07970 DEM:     CALL    0033H           ;DISPLAY CHA
RACTER
4BFC C9        07980          RET                     ;RETURN
4BFD 3E46      07990 FE:      LD      A,'F'           ;LOAD "F" CO
DE
4BFF 18F8      08000          JR      DEM             ;DISPLAY CHA
RACTER
4C01 3E50      08010 PAR:     LD      A,'P'           ;LOAD "P" CO
DE
4C03 18F4      08020          JR      DEM             ;DISPLAY CHA
RACTER
4C05 3EAA      08030 DISB:    LD      A,0AAH          ;LOAD BAR CO
DE
4C07 CD3300    08040          CALL    0033H           ;DISPLAY CHA
RACTER
4C0A 18C5      08050          JR      DISDAT
4C0C 2A        08060 MSG0:    DEFM    '*** EMTROL SYSTEMS, INC. **
*';MESSAGE
4C28 0D        08070          DEFB    0DH
      ;TABLE
4C29 20        08080          DEFM    '       *** LYNX TLS ***'
4C3E 0D        08090          DEFB    0DH
4C3F 00        08100          DEFB    00H
```

Program continues

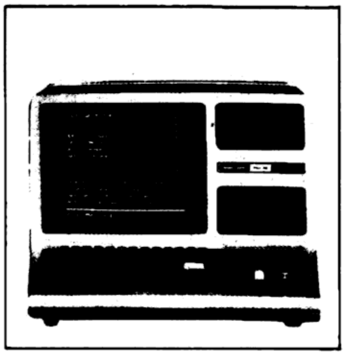
INTRODUCING:

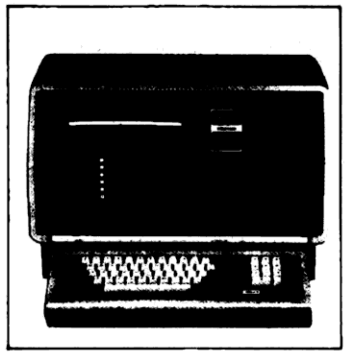
SPECTRONICS, INC.

(A Wholly Owned Subsidiary of Level IV Products)

- OFFERING -

The Full TRS-80* Line at Discount Prices, and Extended Warranty Coverage





COLOR COMPUTER	
16K (R)	$440
16K (E)	$359
16K w/ext basic (R)	$529
16K w/ext basic (E)	$479

MOD III COMPUTER	
16K (R)	$859
48K 2-40 drives (R)	$2010
48K 2-40 drives (E)	$1859
48K 2-80 drives (E)	$2119
RS232 interface add	$89

MOD II COMPUTER	
32K (R)	$2950
64K (R)	$3375
Expansion Dr#1 (R)	$999
Expansion Dr#1-2-3 (E)	$695

NOTE: THE MOD III CAN BE ASSEMBLED WITH UP TO 4 MEG UNFORMATTED FLOPPY DISK STORAGE CAPACITY (4-80 TRK DOUBLE SIDED DRIVES) – CALL FOR A QUOTE ON THE SYSTEM TAILORED TO YOUR EXACT NEEDS.

PRINTERS	
Line Printer IV (R)	$849
Line Printer VI (R)	$986
Daisy Wheel II (R)	$1679
C. ITOH Starwriters	Call
Centronics All	Call
Epson All	Call
Anadex All	Call

DRIVES	COMPLETE	RAW
40 TRK MPI (E)	$309	$245
80 TRK MPI (E)	$429	$360
40/40 TRK MPI (E)	$445	$375
80/80 TRK MPI (E)	$579	$515

PERIPHERALS	
E.I. OK (R)	$249
E.I. 16K (R)	$359
E.I. 32K (R)	$469
E.I. 16K (E)	$305
E.I. 32K (E)	$362
16K RAM (200MS)	$ 39

DEALER'S INQUIRIES INVITED, CALL FOR OUR WHOLESALE DEPARTMENT (313) 525-6201

WARRANTY POLICY

Items including non-Radio Shack components (E) are covered by an extended six-month parts/labor warranty in our tech service center.

Items constructed from Radio Shack approved components (R) are covered by a 90-day parts/labor warranty at any Radio Shack store, plus an additional 90 days in our tech service center.

TERMS OF SALE

2% Discount allowed on cash pre-paid orders. Personal checks require two weeks for bank clearance.

Visa, Master Charge and C.O.D. orders accepted. Add $1.50 for C.O.D.'s.

Shipping/handling/insurance add 1.5% or $2.50 minimum.

Prompt shipments from stock.

SPECTRONICS, INC.
32461 SCHOOLCRAFT ROAD
LIVONIA, MICHIGAN 48150

MICHIGAN – (313) 525-6200
TOLL FREE – 800-521-3305

LEVEL IV PRODUCTS

*TRADEMARK OF TANDY CORP.

See List of Advertisers on page 386

Program continued

```
4C40 0D         08110 MSG1:   DEFB    0DH
4C41 50         08120         DEFM    'PARITY IS '
4C4B 00         08130         DEFB    00H
4C4C 44         08140 MSG2:   DEFM    'DISABLED'
4C54 00         08150         DEFB    00H
4C55 4F         08160 MSG3:   DEFM    'ODD'
4C58 00         08170         DEFB    00H
4C59 45         08180 MSG4:   DEFM    'EVEN'
4C5D 00         08190         DEFB    00H
4C5E 0D         08200 MSG5:   DEFB    0DH
4C5F 43         08210         DEFM    'CHANGE? (Y/N)'
4C6C 0D         08220         DEFB    0DH
4C6D 00         08230         DEFB    00H
4C6E 0D         08240 MSG6:   DEFB    0DH
4C6F 4E         08250         DEFM    'NEW PARITY STATUS? (O/E/N)
4C89 0D         08260         DEFB    0DH
4C8A 00         08270         DEFB    00H
4C8B 0D         08280 MSG7:   DEFB    0DH
4C8C 53         08290         DEFM    'STOP BIT(S) = '
4C9A 00         08300         DEFB    00H
4C9B 0D         08310 MSG8:   DEFB    0DH
4C9C 57         08320         DEFM    'WORD LENGTH = '
4CAA 00         08330         DEFB    00H
4CAB 0D         08340 MSG9:   DEFB    0DH
4CAC 57         08350         DEFM    'WORD LENGTH = (5/6/7/8) BIT
S?'
4CC9 0D         08360         DEFB    0DH
4CCA 00         08370         DEFB    00H
4CCB 0D         08380 MSG10:  DEFB    0DH
4CCC 2A         08390         DEFM    '*** LYNX TLS ***'
4CDC 0D         08400         DEFB    0DH
4CDD 0D         08410 MSG20:  DEFB    0DH
4CDE 53         08420         DEFM    'STORE MESSAGE (S)'
4CEF 0D         08430         DEFB    0DH
4CF0 52         08440         DEFM    'RECEIVE BASIC (R)'
4D01 0D         08450         DEFB    0DH
4D02 54         08460         DEFM    'TRANSMIT BASIC (X)'
4D14 0D         08470         DEFB    0DH
4D15 54         08480         DEFM    'TERMINAL (T)'
4D21 0D         08490         DEFB    0DH
4D22 56         08500         DEFM    'VIEW/CHANGE UART CONFIGURAT
ION (V)'
4D44 0D         08510         DEFB    0DH
4D45 42         08520         DEFM    'BACK TO BASIC (B)'
4D56 0D         08530         DEFB    0DH
4D57 00         08540         DEFB    00H
4D58 49         08550 MSG11:  DEFM    'ILLEGAL ENTRY'
4D65 0D         08560         DEFB    0DH
4D66 00         08570         DEFB    00H
4D67 54         08580 MSG12:  DEFM    'TRANSMIT BASIC (T)'
4D79 0D         08590         DEFB    0DH
4D7A 4C         08600         DEFM    'LOAD PROGRAM (L)'
4D8A 0D         08610         DEFB    0DH
4D8B 00         08620         DEFB    00H
4D8C 0D         08630 MSG13:  DEFB    0DH
4D8D 52         08640         DEFM    'READY CASSETTE'
4D9B 0D         08650         DEFB    0DH
4D9C 48         08660         DEFM    'HIT L TO LOAD'
4DA9 0D         08670         DEFB    0DH
4DAA 00         08680         DEFB    00H
4DAB 53         08690 MSG14:  DEFM    'SENDING PROGRAM'
4DBA 0D         08700         DEFB    0DH
4DBB 00         08710         DEFB    00H
4DBC 52         08720 MSG17:  DEFM    'RECEIVING PROGRAM'
4DCD 0D         08730         DEFB    0DH
4DCE 00         08740         DEFB    00H
0001            08750 HUCR:   DEFS    1                   ;TEMPORARY S
TORAGE
0001            08760 HUSR:   DEFS    1
0001            08770 ERROR:  DEFS    1
0001            08780 PRINT:  DEFS    1
0001            08790 CONTRL: DEFS    1
4DD4 CD2B00     08800 STRBC:  CALL    002BH               ;SUBROUTINE
TO RECEIVE
4DD7 B7         08810         OR      A                   ;BASIC CHARA
CTERS
4DD8 2808       08820         JR      Z,BCR
4DDA FE1B       08830         CP      1BH                 ;IF SHIFT UP
 ARROW, POP
4DDC 2004       08840         JR      NZ,BCR              ;STACK AND R
ETURN TO
4DDE C1         08850         POP     BC                  ;MONITOR MEN
U
```

Program continues

Program continued

```
4DDF C36347     08860           JP      MON
4DE2 DBEA       08870  BCR:     IN      A,(0EAH)            ;WAIT FOR BA
SIC CHARACTER
4DE4 CB7F       08880           BIT     7,A
4DE6 28EC       08890           JR      Z,STRBC
4DE8 DBEB       08900           IN      A,(0EBH)            ;GET BASIC C
HARACTER
4DEA C9         08910           RET
4DEB 0D         08920  MSG18:   DEFB    0DH                 ;MESSAGE TAB
LE
4DEC 53         08930           DEFM    'STORE MESSAGE (S)'
4DFD 0D         08940           DEFB    0DH
4DFE 45         08950           DEFM    'ERASE  MESSAGE (E)'
4E0F 0D         08960           DEFB    0DH
4E10 00         08970           DEFB    00H
4E11 0D         08980  MSG19:   DEFB    0DH
4E12 42         08990           DEFM    'BUFFER FULL'
4E1D 00         09000           DEFB    00H
4E1E DBEA       09010  STRMC:   IN      A,(0EAH)            ;SUBROUTINE
TO RECEIVE
4E20 CB7F       09020           BIT     7,A                 ;MESSAGE CHA
RACTERS
4E22 2803       09030           JR      Z,CFRTMC            ;WAIT FOR CH
ARACTER
4E24 DBEB       09040           IN      A,(0EBH)            ;ECHO
4E26 C9         09050           RET
4E27 CD2B00     09060  CFRTMC:  CALL    002BH               ;SCAN KEYS
4E2A B7         09070           OR      A
4E2B 28F1       09080           JR      Z,STRMC             ;IF NO ENTRY
, CONTINUE
4E2D FE1B       09090           CP      1BH                 ;IF SHIFT UP
 ARROW, TO
4E2F 20ED       09100           JR      NZ,STRMC            ;MENU, BUT F
IRST POP
4E31 C1         09110           POP     BC                  ;STACK
4E32 C36347     09120           JP      MON                 ;TO MONITOR
MENU
4590            09130           END     RELSRT
```


✓See List of Advertisers on page 386

The students meet the POKE instruction.

Alternate Course—Part III

Michael A. Duffin
1507 East Avenue
Berwyn, IL 60402

In most data processing courses, the student usually does not write any programs which might affect the system—especially a large system. Many large systems have enough trouble running without novice programmers experimenting with them. The TRS-80, on the other hand, is a one-on-one system, so the only person that is hurt by this type of experimentation is the programmer.

In my data processing class, I introduced the POKE instruction to illustrate graphics. There are three reasons I chose the POKE instruction rather than Set and Reset.

First, when using POKE for graphic representations the first operand must be a number between 15360 and 16383. These are the positions in memory (decimal) which control what is on the screen (i.e. POKEd characters in these positions are reflected on the screen). POKEing outside of this range will result in a variety of errors and in many cases Memory Size? will appear on the screen. When this occurs the user's program is lost. By usine POKE my students learn what is stored in memory by the operating system.

Second, an additional operand of the POKE instruction is the ASCII (American Standard Code for Information Interchange) representation of a character. Thus, when we talked about ASCII in class, the students had a concrete example to refer to.

Third, when using the POKE instruction, six bits are set on the screen at once. With the Set instruction only one bit is set at a time. By the way, when you program a figure from a TRS-80 video display worksheet using POKE you must add 15360 to the numbers 0 through 1023 so the figure will appear on the corresponding positions on the screen.

Before we could begin representing graphics on the screen using POKE, the first thing we had to do was determine which bit position combinations corresponded to what ASCII numbers. All the combinations are represented by ASCII numbers 129 through 191. I gave my students the following program to determine the correspondence between numbers and characters:

```
10  CLS
20  FOR A = 129 to 191
30  PRINT A " = " CHR$(A)
40  PRINT
50  NEXT A
```

I also gave them the following instructions with this program:

- Type this program in and run it.
- Add the necessary code to this program to slow it down so the character representations can be recorded.
- Record the character representations of 129 to 191 on a TRS-80 video display worksheet.
- Describe how this program works line by line.

The problem with this program is the characters flash by too fast to be recorded. If we add the lines,

```
42  FOR X = 1 to 250
44  NEXT X
```

the For...Next loop slows the program enough to record the characters. Should more time be needed, hit the Break key to stop the program. To continue, type in CONT and hit the Enter key.

In this program, line 10 clears the screen; lines 20 and 50 create a loop starting with the number 129 and ending with the number 191. Line 40 prints a blank line between each character representation. Line 30 prints the current numeric value of A followed by an equal sign. The instruction CHR$(A) prints the ASCII equivalent of the number A. For this problem all the ASCII equivalents are graphic characters.

Once my students had recorded all the graphic characters, I gave them the following program to type in:

```
10  CLS
20  FOR X = 15362 to 16376 STEP2
30  POKE X - 2, 32
40  POKE X - 1, 32
50  POKE X, 32
60  POKE X + 1, 32
70  POKE X, 176
80  POKE X + 1, 190
90  POKE X + 2, 189
100 POKE X + 3, 176
110 NEXT X
120 GOTO 20
```

I then gave my students the following instructions:

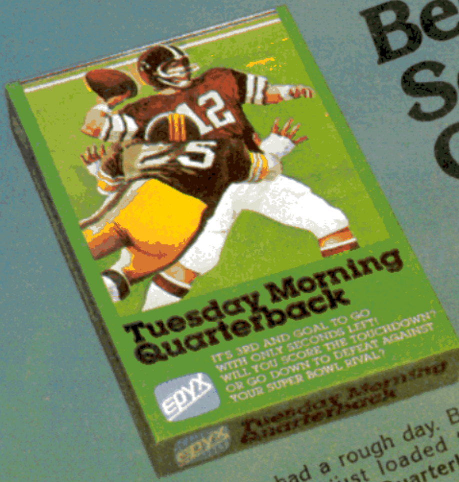
- Enter this program and run it.
- Describe how the program works line by line.
- Remove the words STEP2 from line 20. How does this affect the program?
- Write a program that will cause the object to move in a different direction.

In the program a small flying saucer moves across the screen from left to right, top to bottom. The saucer is continuously erased and rebuilt so it appears to move across the screen. More specifically, the For. . . Next loop in lines 20 and 110 cause X to be incremented from 15361 to 16379. STEP2 causes this incrementation to be done be twos (i.e. 15362, 15364, 15366, etc.). Removal of STEP2 causes the saucer to move half as fast. Lines 30 through 60 erase the saucer. (i.e. the ASCII number 32 causes a six-bit graphic character to be erased). Lines 70 through 100 rebuild the saucer. This program is a little unusual because it starts with a non-existent figure being erased and then builds a figure. X is then increased by two, the previous figure is erased, the next figure is built, and so on, until X is greater than 16377.

As a further exercise I asked my students to write a program which would cause the saucer to move in three other directions. This isn't that simple because the sample program is not geared for many other directions. For example, if we wish the saucer to move from right to left, the For. . . Next loop is of little use in its present form. Thus, they first had to determine a set-up for the program that could be adaptable for any direction. The right to left example that follows illustrates this:

```
10 CLS
20 X = 16379
30 GOSUB 1000
40 X = X - 1
50 IF X<15360 THEN GOTO 20
```


Reader Service for facing page ✓48



Beat The Socks Off Your Computer's Quarterback—You Hope

... You've had a rough day. But now it's fun time; you've just loaded the brand new **Tuesday Morning Quarterback**™ program into your computer. You're the quarterback. And the coach, general manager, star. You're going to beat the socks off your computer. Or off a human opponent. You hope.

You pick the NFL team that plays closest to your management style. Now you assign a team to your computer ... or let a friend pick out an opposing team. If you want a better chance to win, pick a strong team; give your computer a weak team. (Don't try giving a weak team to a friend; just make sure you get first choice. After all, it's your game.)

You still have to play smart, though. And have a bit of luck on your side, too. Otherwise, your computer might beat the socks off **you.** (Later, as you get better, you can show your smarts by having **your** weaker team beat your computer's stronger team.

Now it's game time. List your line-up. If it's your ball, call the plays: running plays like quarterback sneak*, draw*, sweep* ... 6 choices. Or pass plays: short curl*, screen*, bomb* ... 8 choices. And option* and trick plays, too.

The opponent has the ball. Call a normal 4-3 or 3-4 defense*. Or pull a line-backer out for a nickel* pass defense. Or call for the blitz*; you figure the other team is going to pass and you'd like to sack* their quarterback. But if their team elects to run, you're probably in trouble. 6 defensive strategies, including the Butkus*.

Tuesday Morning Quarterback. 16 offensive plays; 6 defensive strategies. And rosters to pick from, draft choices and trades to make. With a probability structure that comes unbelievably close to the real thing. It's **never** the same game twice. Superb graphics, real-time. Even **color** (with the Apple) and sound. You won't get bored playing the game. Not in your or your computer's lifetime.

Another bug-free, easy-loading lifetime computer game from EPYX. With the unique **EPYX lifetime warranty:** If anything happens to your cassette or disk at any time and for any reason, send it back with just $5.00 for shipping and handling and we'll send you a brand new one.

(Of course, there is also our 30-day unconditional guarantee: If your EPYX game has any defect whatsoever within 30 days of purchase, return it to us or to your dealer and we will replace it free.)

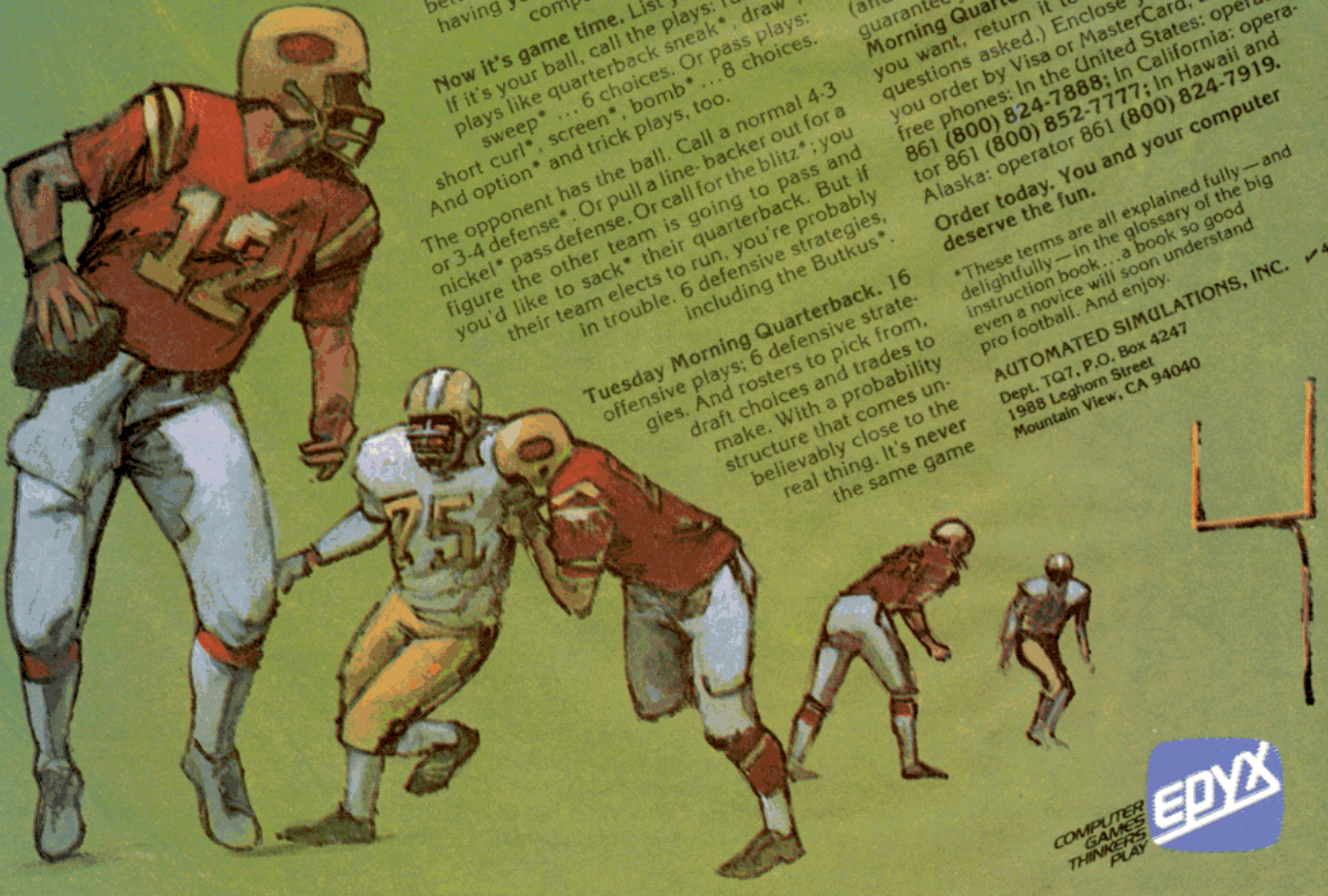
Visit your dealer now and pick up **Tuesday Morning Quarterback** in its good-looking, protective box with the best instruction book you ever read. Now available on disk for the Apple II (48K RAM with Applesoft) and Radio Shack's TRS 80 (48K RAM). **Only $29.95.**

If your dealer is out of stock and you can't wait, order directly from Automated Simulations. **$29.95** plus $2.00 for shipping & handling (and sales tax if you're in California). We guarantee you will be delighted with **Tuesday Morning Quarterback.** (If it isn't exactly what you want, return it to us for full refund. No questions asked.) Enclose your check. Or if you order by Visa or MasterCard, use our toll-free phones: In the United States: operator **861 (800) 824-7888;** In California: operator **861 (800) 852-7777;** In Hawaii and Alaska: operator **861 (800) 824-7919.**

Order today. You and your computer deserve the fun.

*These terms are all explained fully—and delightfully—in the glossary of the big instruction book ... a book so good even a novice will soon understand pro football. And enjoy.

AUTOMATED SIMULATIONS, INC. ✓48
Dept. TQ7, P.O. Box 4247
1988 Leghorn Street
Mountain View, CA 94040



EPYX
COMPUTER GAMES THINKERS PLAY

". . . the only person hurt by this type of experimentation is the programmer."

```
60 GOTO 30
1000 POKE X,176
1020 POKE X+1,190
1030 POKE X+2,189
1040 POKE X+3,176
1050 POKE X,32
1060 POKE X+1,32
1070 POKE X+2,32
1080 POKE X+3,32
2000 RETURN
```

In this program the subroutine at lines 1000 through 2000 can be used no matter what direction the figure moves. The key to the movement is in lines 40 and 50. Line 40 decrements the value of X by one, and thus the saucer appears to move from right to left. Line 50 is necessary to keep the ship from going off the screen, and it also restarts the program when it reaches the end of the screen.

If we wish to move in another direction we only have to change lines 40 and 50. For example, if we wish to move from top to bottom in a left to right manner we would use the following:

```
40 X=X-64
42 IF X+3>16383 THEN X=X-1023
50 IF X+3=16383 THEN X=15360
```

Line 40 writes X in the position directly below the previous X. Line 42 places the figure at the top of the screen after the figure has been printed at the bottom. The figure at the top is printed to the right of the previous figure at the bottom. If we wish the figure to be printed in a top to bottom, right to left manner, we need only substitute the following for line 42:

```
IF X+3>16383 THEN X=X-1025
```

Line 50 prints the figure at the upper left of the screen before if goes past the bottom right of the screen.

Now that we can get the figure to move in any direction, it would be nice if we could switch directions at the push of a button. To do this we have to make use of the INKEY$ instruction. As an introduction, I gave my students the following program and instructions:

```
10 CLS
100 PRINT 10:GOSUB 1000
110 GOTO 100
200 PRINT 20:GOSUB 1000
210 GOTO 200
300 PRINT 30:GOSUB 1000
310 GOTO 300
400 PRINT 40:GOSUB 1000
410 GOTO 400
500 PRINT 50:GOSUB 1000
510 GOTO 500
1000 FOR X=1 TO 200;NEXT X
1010 IF INKEY$ = "B" THEN GOTO 200
1100 RETURN
```

- Run the above program and hit the letter B. What happens?
- Modify the program above so that lines 100, 200, 300, 400 or 500 get executed when a different letter is pressed.
- What does INKEY$ do?

When the program is executed, the number 10 is written periodically until the letter B is hit. At this time the number 20 is printed.

Modifying the program is somewhat difficult. Due to a peculiarity of the INKEY$ instruction, once it is accessed the value disappears. Thus, if we just used If statements between lines 1000 and 1100, only the first If statement would reflect the key that was depressed. If we want to check the depressed key more than once, we must use code similar to the following:

```
1010 B$ = INKEY$
1020 IF B$ = "A" THEN GOTO 100
1030 IF B$ = "B" THEN GOTO 200
1040 IF B$ = "C" THEN GOTO 300
1050 IF B$ = "D" THEN GOTO 400
1060 IF B$ = "E" THEN GOTO 500
```

By storing the value of INKEY$ within our program, we can access it as often as we like without it disappearing.

But now we need an If statement for every check we wish to make. To avoid this I introduced my students to the ON . . . GOTO instruction with the following changes in the program:

```
1000 FOR X=1 TO 200:NEXT X
1010 ON VAL(INKEY$) GOTO 100,200,300,400,500
1020 RETURN
```

The way the ON . . . GOTO instruction works is more clearly explained if I use the following example:

```
ON X GOTO 100,500,200
```

If X = 1 then we go to line 100. If X = 2 then we go to line 500. If X = 3 then we go to line 200. If X has any other value, the next statement in the program is executed. In line 1010 we must use the VAL instruction because INKEY$ gives us the ASCII equivalent of the number we have entered. By using VAL the ASCII characters are converted to numerics.

Now that we can move our figure in any direction by hitting a key, there are a few other things we might desire if we want to make this program into a game. First, we might want more than one figure on the screen. Second, we might want them to shoot at each other; and finally we must decide if the second figure will be controlled by another person or the computer.

The 13-line subroutine in Fig. 1 constructs a second figure whose movement is controlled by hitting the letters A,B,C or D. The values given to the W array in conjunction with lines 3020 through 3050 and 3100 control the direction of the figure.

Line 3020 causes the letters A through D to be converted to the numbers one through four. Line 3020 subtracts 64 from the ASCII equivalent of the character depressed while the program is running. Since the ASCII equivalents of A,B,C and D are 65,66,67 and 68 respectively, this line gives variable B a value of one when A is depressed, two when B is depressed, three when C is depressed and so on. Line 3030 allows only A through D to be depressed; all other letters set B to equal one. Line 3040 erases the figure.

Line 3050 causes Y to be incremented by a member of the array W. The values of this array were established in line 12 along with the initial values of Y and B. The variable B points to the correct element.

To illustrate this, let us assume that C was depressed while Y = 16005. Then line 3020 becomes:

```
B=ASC(C)-64   which means
B=67-64       which means
B=3.
```

The If statement in line 3030 fails since B = three. Line 3040 erases the figure that started at Y = 16005.

Line 3050 translates to:

```
Y=16005 + W(3) since W(3)=64 from line 12
Y=16005 + 64
Y=16069.
```

The If statements in lines 3060 through 3090 re-establish Y if the value of Y extends beyond the range of the screen (i.e. 15360 to 16383). Line 3100 then builds the new figure at the desired location.

To put it more simply, if A is entered, the figure moves from right to left. If C is entered the figure goes down, and if D is entered the figure moves up.

Fig. 2 illustrates another version of this subroutine. In this version the computer controls the ship (variable Y). The computer's ship chases another ship (variable X). In addition, the computer's ship fires upon the user's ship (lines 3130 through 3200). With minor modifications both of these routines can be included within the same program. I suggest changing Fig.1 in the following manner:

- Change the line numbers.
- Change variable Y to X.
- Change Y = 16379 to X = 15360 in line 12.
- Change the variable B to another variable name (such as C).

By adding a couple of GOSUBS you'll have one ship chasing another. However, since the user's ship doesn't shoot, you had better add a routine for this if you intend to win.■

How's your love life?
INTERLUDE #99 THE ULTIMATE EXPERIENCE...
A little dull around the edges? Routine? Predictable? Boring? Maybe all it needs is a little Interlude. Interlude is the most stimulating computer game ever conceived. It combines a computer interview, an innovative programming concept, and a one-of-a-kind manual to turn your love life into exciting, adventurous, delicious fun!
Interlude is: romantic…playful…outrageous…a fantasy. Interlude is: ■ A Bed of Roses (Interlude #1) ■ Mata Hari (Interlude #49) ■ The Chase (Interlude #7) ■ Rodeo! (Interlude #71) ■ The King and I (Interlude #60) ■ Some Enchanted Evening (Interlude #84) ■ Caveman Caper (Interlude #82) ■ From Here to Ecstasy (Interlude No. 30) ■ Satin Dreams (Interlude #72).
More than 100 Interludes are included in the program. Most are described in detail in the accompanying manual, but several surprise Interludes are buried in the program awaiting that very special time when your interview says you're ready. (When you learn secret Interlude #99, your love life may never again be the same!) Interlude can give you experiences you'll never forget. Are you ready for it?
287
Interlude™
The Ultimate Experience.
INTERLUDE, Dept. 80, 10635 Richmond Avenue, Houston, Texas 77042. I'm really ready. Send my Interlude today.
Apple II (16K)* ☐ Cassette ($18.95) ☐ Diskette ($21.95) ☐ Diskette — Pascal or DOS 3.3 ($21.95)
TRS-80 (Model 1, 16K)** ☐ Cassette ($18.95) ☐ Diskette ($21.95)
Add $1.75 for shipping and handling.
Poster ☐ 20"x 24" reproduction of this ad without ad copy ($4.95—includes shipping charges)
Available for immediate shipment.
Please enclose your check payable to INTERLUDE or complete the charge information:
☐ MASTERCARD ☐ VISA All charge customers must sign here.
Account No. Expiration date MasterCard Bank Code
CHARGE CUSTOMERS: Order by phone toll-free! 1-800-231-5768 Ext. 306 (Texas: 1-800-392-2348 Ext. 306)
Name Age
Address
City State Zip
*Apple II is a registered trademark of Apple Computers, Inc. **TRS-80 is a registered trademark of Radio Shack, a Tandy Co.

This program will lay-up your TRS-80 for more than three seconds.

Hoops

William Cornwell
609 Ellynn Drive
Cary, NC 27511

"Score! Doctor J adds two more!", exclaimed the television announcer. "Dad, how many points does Doctor J have now?" asked my 14-year old son. I couldn't tell him as I wasn't keeping score myself, but as I glanced up with my response, I noticed our Level II 16K TRS-80 nearby, but not in operation.

What a great application—design a basketball scorekeeping program. So in the next few evenings I thought considerably about what would be required.

The more I thought, the better the program appeared. First, I located a video display worksheet and penciled in the format of the output—team names, scores and fouls. I then provided lines for the individual players—numbers, names, number of field goals and fouls, total points scored, and number of personal fouls (see Photo 1). I discovered that a two-dimensional, 14-member array in Basic would handle all of this data nicely.

With the output out of the way, I next considered the input data. One goal was to have the ability to correct any character keyed in error prior to entering the data, or provide a means of ignoring the input request from any character position. Based on my Basic experience, this precluded the use of an INKEY$ type of routine. Also, I wanted any input combination to be one character string so the user would not have to continually key additional commas as field separators. This would also provide the flexibility of having one, two, and four-character input coding strings.

Next, I started to code the Basic program logic, but on paper first! This turned out to be well worth the effort as I was able to develop the total program in a subroutine fashion—optimizing statements as I coded—and using open boxes for GOTO label numbers. When the Basic coding was finished, I wrote the statement numbers down for both the Basic statements and then in the GOTO boxes.

Since I don't have the luxury of a printer, the paper draft provided outstanding reference material for debugging and cosmetic changes. And I made several of each.

The moment of truth arrived as I keyed Run. Concept was good, but I had to make several other minor adjustments for better viewing on the output display. I was quite happy with the results—the paper draft netted a nearly bug-free implementation! Numerical column alignment using Basic with combinations of single and double-digit numbers was the toughest problem. Screen data is presented in anywhere from one point five to three seconds depending upon the number of players.

After loading the program and keying Run to begin program initiation, the user is questioned whether or not he desires an explanation of the program and the instructions. Assuming the user wants this information, he keys a Y (for yes) and a description of the program is displayed.

He is then requested to push the Enter key to continue program operation. The Help screen (see Photo 2) is displayed. The next logical function would be to load both a home team and a visiting team. Com-

Photo 1. Sample Output Reflecting Box Score of NBA final Playoff Game Box Score (Player Fouls are Estimated).

"What a great application— design a basketball scorekeeping program."

mands for each appear later.

The Enter Data? request appears at the bottom line of the screen as a result of: keying N (for no) to the information request during program initiation; pushing Enter after viewing the program information and input coding structure display; the successful conclusion of a previously keyed input command.

Since the Enter Data? request is where all the action occurs during the course of this program, a more detailed description of the input code structure follows. After keying any input code, the user pushes the Enter key to initiate program action for that code.

Valid single-character entries are:

?—I have a question and need help—this will display the input coding structure (the Help screen in Photo 2) that is acceptable to the Enter Data? request.
D—Display—displays the team and player totals.
X—Input keying error—this character appearing in any one of the four positions of the input code positions signals an input error and the user can respond to another Enter Data? request.
E—Exit—exits or terminates the program.

Valid two-character entries are:

HA—Add player(s) to home team—user must enter a two-character number, push the Enter key, and then key a name of 12 characters or less and push the Enter key again. If no players' names exist, the user will be requested to key a team name with the same 12 characters or less limitation followed by the Enter key prior to entering any player information.
VA—Add player(s) to visiting team—same description as the HA code except information applies to the visiting team.

Valid four-character entries are: * indicates that the first character will be either an H for a home team player or a V to indicate a visiting team player; the second character indicates what event occurred;

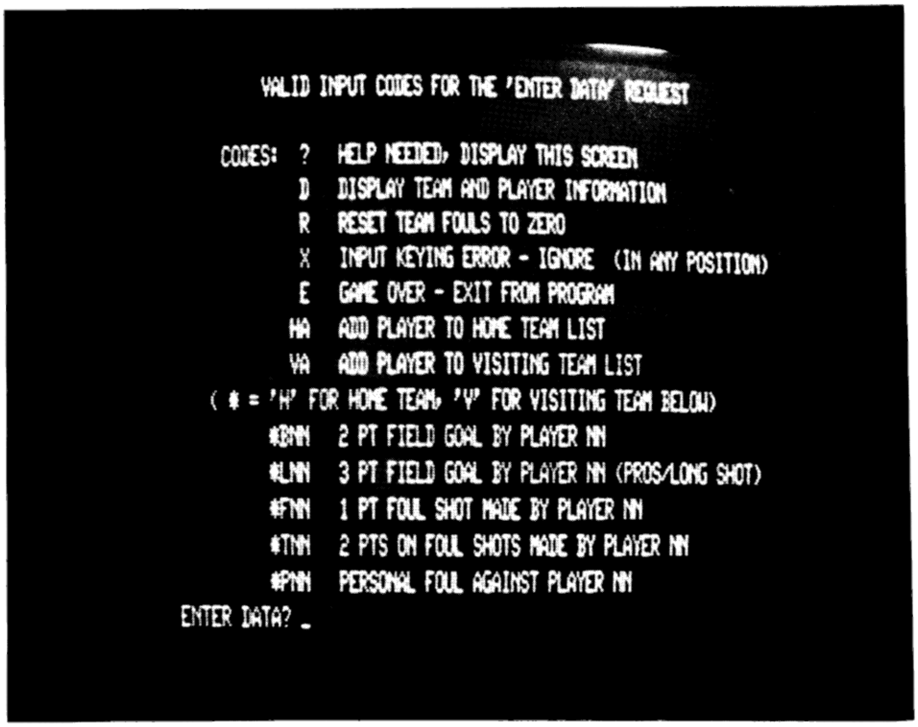


Photo 2. Sample of Help Screen Display.

"The program provides the basics for basketball scorekeeping and was a joy to develop."

the last two characters of NN represent the number of the player.

- *BNN—Basket scored—player NN scored a two-point field goal.
- *LNN—Long basket scored—player NN scored a three-point field goal.
- *FNN—Foul shot successful—player NN scored one successful free throw.
- *TNN—Two foul shots successful—player NN scored two successful free throws.
- *PNN—Personal foul—player NN has committed a personal foul.

The Help screen (see Photo 2) provides a concise listing for the above codes.

For those readers who do not care about the internal data structure or have no intention of modifying the program, they should conclude with the summary paragraphs. As I stated earlier, the data area for the program is a two-level, 14-member array for each field: player's number (A$), player's name (B$), number of field goals (J), number of successful foul shots (K), total points (L) and number of personal fouls (M).

Variables N and P are used for updating the array fields: N is the level value—zero for the home team and a one for the visiting team. This value is determined from the first character of the input coding structure. The variable P, when it ranges from one to 13, represents a player value. When this variable is a zero it refers to a team value.

For instance, the name field, B$ (0,0) or B$ (1,0), refers to the team names. The field goal variable, J (0,0) or J (1,0), contains the number of players currently entered in the team list (see Fig. 1). The total points field for the team contains the team score, and the number of personal fouls field contains the number of team fouls. But this last value never exceeds seven for either team since seven will handle the foul bonus situations for all levels of basketball.

In summarizing, I wanted a useful, simple and meaningful code structure. The basketball scorekeeping program, I think, meets these original objectives. In addition, the program, as is, applies to any level of basketball—professional, collegiate, scholastic, or non-scholastic. The program can also be used with several of the commercial basketball games available such as Basket.

The program provides the basics for basketball scorekeeping and was a joy to develop. Some possible enhancements include: Loading the players' names and numbers via tape, disk or data statements; printing a final box score at the game's conclusion; saving player and team totals on some media for use with additional averaging-type programs; and, lastly, correcting any error once entered (points credited to the wrong player, disallowed baskets or fouls). I would be interested in copies of any such enhancements. ■

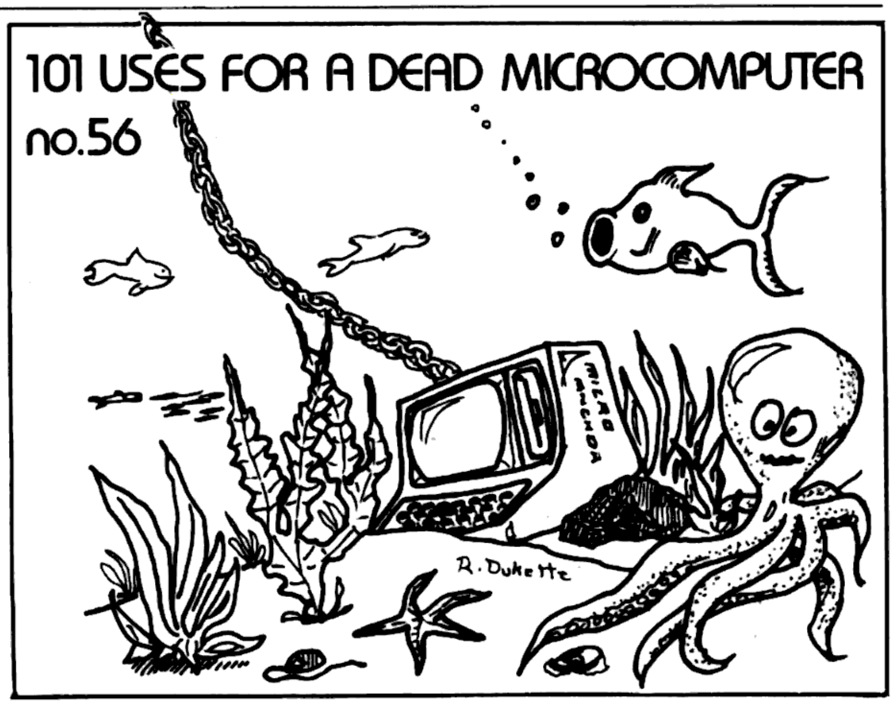
Field	Variable	Team information	Player information	
		P = 0	P = 1	P = 13
Player's number	A$	—Not used—	Player's number	Player's number
Player's name	B$	Team name	Player's name	Player's name
Field goals made	J	Number of players on team	Player's field goals	Player's field goals
Fouls made	K	—Not used—	Player's fouls made	Player's fouls made
Total points	L	Team point total	Player's point point total	Player's point point total
Personal fouls	M	Team personal	Player's personal fouls	Player's personal fouls

Fig. 1. Summary of Array Information (Home Team when N = 0, visiting team when N = 1).

Program Listing

```
10 REM INITIALIZE AND SET MAXIMUM PLAYER LIMIT (P=13)
20 CLEAR 1000:P=13
30 DIMA$(1,P):DIMB$(1,P):DIMJ(1,P):DIMI$(2):DIMK(1,P):DIML(1,P):
DIMM(1,P)
40 CLS:PRINT@144,"BASKETBALL SCOREKEEPING PROGRAM"
```

Program continues



Program continued

```
50 PRINT@351,"BY"
60 PRINT@536,"wILLIAM cORNWELL"
70 PRINT@600,"609 ELLYNN DRIVE"
80 PRINT@664,"CARY, N. C. 27511"
90 PRINT:PRINT
100 REM TEST TO SEE IF INFORMATION WANTED
110 INPUT "INSTRUCTIONS NEEDED?  KEY: Y/N";X$:CLS
120 IF X$="N" THEN 180
130 REM GO SHOW THE INFORMATION SCREEN
140 GOSUB 1130
150 REM GO SHOW THE VALID CODE SCREEN
160 GOSUB 1270
170 REM MAIN PROGRAM LOGIC - ROUTINE RETURN POINT
180 X$="     "
190 PRINT@961,"ENTER DATA";:INPUT X$
200 REM SET AND VALIDATE LENGTH OF RESPONSE
210 N=LEN(X$):IF N=3 OR N>4 THEN PRINT@1000,"INPUT NOT 1,2 OR 4
CHARS";:GOTO180
220 REM SEPARATE FIRST CHARACTER OF RESPONSE
230 I$(1)=LEFT$(X$,1)
240 IF I$(1) = "X" THEN 180
250 IF N=1 THEN 340
260 REM SEPARATE SECOND CHARACTER OF RESPONSE
270 I$(2)=MID$(X$,2,1)
280 IF I$(2) = "X" THEN 180
290 REM AND GET THE FINAL TWO RESPONSE CHARACTERS
300 IF N>2 THEN I$(0)=MID$(X$,3,2)
310 IF MID$(X$,3,1) = "X" THEN 180
320 IF MID$(X$,4,1) = "X" THEN 180
330 REM VALIDATE FIRST RESPONSE CHARACTER
340 IF I$(1)="D" THEN 1050
350 IF I$(1) <> "H" THEN 380
360 REM HOME TEAM SET UP ARRAY VALUE
370 N=0:GOTO 540
380 IF I$(1) <> "V" THEN 410
390 REM VISITING TEAM SET UP ARRAY VALUE
400 N=1:GOTO 540
410 IF I$(1) <> "R" THEN 440
420 REM RESET TEAM FOUL ROUTINE
430 M(0,0)=0:M(1,0)=0:PRINT@1000,"TEAM FOULS RESET";:GOTO 180
440 IF I$(1)="X"THEN 180
450 REM SEE IF USER WANTS HELP
460 IF I$(1)="?" THEN GOSUB 1270 :GOTO180
470 REM  USER KEY / IN ERROR?
480 IF I$(1)="/" THEN GOSUB1270 :GOTO180
490 REM USER WANT TO EXIT PROGRAM
500 IF I$(1)="E" THEN END
510 REM CODE NOT VALID - GIVE USER ANOTHER TRY
520 GOTO 180
530 REM BEGIN SECOND CHARACTER VALIDATION
540 IF I$(2)="A" THEN 630
550 REM  ADD TEAM/PLAYER ROUTINE  H OR V SET UPON ENTRY
560 REM EXTRACT VALUE FOR NO OF PLAYERS
570 Q=J(N,0)
580 FOR P=1 TO Q
590  IF I$(0)=A$(N,P) THEN 810
600 NEXT
610 REM PLAYER NOT IN LIST
620 PRINT@1000,"PLAYER NOT FOUND";:GOTO 180
630 CLS:Q=J(N,0)
640 IF Q<>0 THEN 680
650 REM NO PLAYERS YET - REQUEST TEAM NAME
660 INPUT "ENTER TEAM NAME";B$(N,0)
670 K=LEN(B$(N,0)):IF K>12 THEN PRINT "TEAM NAME 12 CHARS OR LES
S - REENTER":K=0:GOTO 660
680 Q=Q+1
690 IF Q<14 THEN 720
700 PRINT@1000,"TOO MANY PLAYERS";:INPUT C$:GOTO 1050
710 REM ADD PLAYER ROUTINE
720 INPUT "ENTER PLAYER'S NUMBER AS TWO DIGITS (05,13)";A$(N,Q)
730 K=LEN(A$(N,Q)):IF K<>2 THEN PRINT "NO MUST BE 2 DIGITS - REE
NTER":K=0:GOTO 720
740 INPUT "ENTER PLAYER'S NAME (12 CHAR LIMIT)";B$(N,Q)
750 K=LEN(B$(N,Q)):IF K>12 THEN PRINT "PLAYER'S NAME MUST BE 12
CHARS OR LESS - REENTER":K=0:GOTO 740
760 INPUT "MORE PLAYER'S TO ADD?  KEY: Y/N";Z$
770 IF Z$="N" THEN 800
780 Q=Q+1:PRINT:GOTO 720
790 REM SAVE NO OF PLAYERS AND EXIT
800 J(N,0)=Q:GOTO 1050
810 IF I$(2)<>"B" THEN 850
820 REM BASKET SCORED ROUTINE
830 J(N,P)=J(N,P)+1
```

Program continues

Program continued

```
840 L(N,P)=L(N,P)+2:L(N,0)=L(N,0)+2:GOTO 1050
850 IF I$(2) <>"L" THEN 890
860 REM 3 POINT BASKET SCORED ROUTINE
870 J(N,P)=J(N,P)+1
880 L(N,P)=L(N,P)+3:L(N,0)=L(N,0)+3:GOTO 1050
890 IF I$(2)<>"T" THEN 930
900 REM TWO FOULS MADE ROUTINE
910 K(N,P)=K(N,P)+2
920 L(N,P)=L(N,P)+2:L(N,0)=L(N,0)+2:GOTO 1050
930 IF I$(2)<>"F" THEN 970
940 REM ONE FOUL SCORED ROUTINE
950 K(N,P)=K(N,P)+1
960 L(N,P)=L(N,P)+1:L(N,0)=L(N,0)+1:GOTO 1050
970 IF I$(2)<>"P" THEN 1030
980 REM PERSONAL EOUL ROUTINE
990 IF M(N,P)<6 THEN M(N,P) = M(N,P)+1
1000 IF M(N,0)<7 THEN M(N,0)=M(N,0)+1
1010 GOTO 1050
1020 REM  SECOND CHARACTER INVALID - TELL USER
1030 PRINT@1000,"SECOND CHARACTER INVALID";:GOTO 180
1040 REM SCREEN DISPLAY ROUTINE
1050 CLS:N=0:P=1:I=J(0,0):IF J(1,0)>I THEN I=J(1,0)
1060 PRINT TAB(0)"H:";TAB(3)B$(N,0);" ->";L(M,0);TAB(24)"TF-";M(
N,0);TAB(33)"V:";TAB(36)B$(P,0);" ->";L(P,0);TAB(57)"TF-";M(P,0)
1070 PRINT TAB(1)"# PLAYER";TAB(17)"G   F  PT  PF";TAB(34)"# PLA
YER";TAB(50)"G   F  PT  PF"
1080 FOR Q = 1 TO 13
1090 IF I<Q THEN PRINT:GOTO 1110
1105 PRINT TAB(0)A$(N,Q);TAB(3)B$(N,Q);TAB(16)J(N,Q);STR$(K(N,Q)
);TAB(23)STR$(L(N,Q));TAB(27)STR$(M(N,Q));TAB(33)A$(P,Q);TAB(36)
B$(P,Q);TAB(49)J(P,Q);STR$(K(P,Q));TAB(56)STR$(L(P,Q));TAB(60)ST
R$(M(P,Q))
1110 NEXT
1120 GOTO 180
1130 REM INFORMATION SCREEN DISPLAY ROUTINE
1140 CLS:PRINT TAB(16)"BASKETBALL SCOREKEEPING PROGRAM"
1150 PRINT:PRINT"PROGRAM WAS DESIGNED FOR EASE OF USE WHILE KEEP
ING SCORE OF A"
1160 PRINT"BASKETBALL GAME IN PROGRESS AT ANY COMPETITIVE LEVEL
-- PRO,"
1170 PRINT"COLLEGE, SCHOLASTIC OR PICK-UP VARIETY.":PRINT
1180 PRINT"PROGRAM FEATURES A 'HELP' FACILITY BY KEYING '?' AND
CODES FOR:"
1190 PRINT"2 POINT FIELD GOALS, 3 POINT FIELD GOALS (PROS), 1 OR
 2 FOUL"
1200 PRINT"SHOTS MADE, AND RECORDING A PLAYER'S PERSONAL FOULS."
:PRINT
1210 PRINT"LIMITATIONS INCLUDE: 12 CHARACTER MAXIMUM FOR NAMES O
F TEAMS"
1220 PRINT"AND PLAYERS, TEAMS CAN HAVE A MAXIMUM OF 13 PLAYERS."

1230 PRINT:PRINT
1240 PRINT TAB(39)"PUSH 'ENTER' TO CONTINUE"
1250 INPUT C$
1260 RETURN
1270 REM VALID INPUT CODE SCREEN DISPLAY ROUTINE
1280 CLS:PRINT@9,"VALID INPUT CODES FOR THE 'ENTER DATA' REQUEST
"
1290 PRINT@133,"CODES:  ?":PRINT@145,"HELP NEEDED, DISPLAY THIS
SCREEN"
1300 PRINT@205,"D":PRINT@209,"DISPLAY TEAM AND PLAYER INFORMATIO
N"
1310 PRINT@269,"R":PRINT@273,"RESET TEAM FOULS TO ZERO"
1320 PRINT@333,"X":PRINT@337,"INPUT KEYING ERROR - IGNORE  (IN A
NY POSITION)"
1330 PRINT@397,"E":PRINT@401,"GAME OVER - EXIT FROM PROGRAM"
1340 PRINT@460,"HA":PRINT@465,"ADD PLAYER TO HOME TEAM LIST"
1350 PRINT@524,"VA":PRINT@529,"ADD PLAYER TO VISITING TEAM LIST"

1360 PRINT@580,"( * = 'H' FOR HOME TEAM, 'V' FOR VISITING TEAM B
ELOW)"
1370 PRINT@650,"*BNN":PRINT@657,"2 PT FIELD GOAL BY PLAYER NN"
1380 PRINT@714,"*LNN":PRINT@721,"3 PT FIELD GOAL BY PLAYER NN (P
ROS/LONG SHOT)"
1390 PRINT@778,"*FNN":PRINT@785,"1 PT FOUL SHOT MADE BY PLAYER N
N"
1400 PRINT@842,"*TNN":PRINT@849,"2 PTS ON FOUL SHOTS MADE BY PLA
YER NN"
1410 PRINT@906,"*PNN":PRINT@913,"PERSONAL FOUL AGAINST PLAYER NN
"
1420 RETURN
1430 PRINT@998,"PUSH 'ENTER' TO CONTINUE"
1440 INPUT C$:RETURN
```

Continued from page 117

"To help us recognize the 'programs' or levels of thought in our own minds, Hofstader engages us in various mind games."

ple. There is no psychological proof to chart the metamorphosis of neuron action to symbol recognition to iteration. To help us recognize the "programs" or levels of thought in our own minds, Hofstader engages us in various mind games. This is the purpose of the fictional dialogs that follow each chapter of *Godel, Escher, Bach*.

The main characters of the dialogs are the Tortoise and Achilles. The choice of these figures was inspired by Zeno and by Lewis Carroll. Zeno, a Greek mathematician, wrote a tale about a foot race between the Tortoise and Achilles to illustrate his theories of paradoxes of motion. Lewis Carroll, the author of *Alice in Wonderland* and a Nineteenth-century mathematician, borrowed the Tortoise and Achilles from Zeno to write a dialog about reasoning Carroll's dialog is reprinted in *Godel, Escher, Bach*. Each dialog in the book helps us conceptualize the intricate methods of human thought.

Hofstader's dialog titled "Crab Canon" tells the story of a chance meeting between Achilles and the Tortoise in the park. In the middle of their conversation, they are interrupted by a friend, the Crab. The title, "Crab Canon," has more significance in relation to the dialog.

"Crab Canon", you'll recall, is a theme followed by itself in reverse. The notes read the same melody backwards and forwards. In Hofstader's "Crab Canon" the lines are virtually the same when they are read from the end or from the beginning. For instance, the opening lines of the dialog are:

Tortoise: Good day, Mr. A.
Achilles: Why, same to you.
Tortoise: So nice to run into you.
Achilles: That echoes my thoughts.
Tortoise: And it's a perfect day for a walk. I think I'll be walking home soon.

And the closing lines:

Achilles: And it's a perfect day for a walk. I think I'll be walking home soon.
Tortoise: That echoes my thoughts.
Achilles: So nice to run into you.
Tortoise: Why, same to you.
Achilles: Good day, Mr. T.

When the Crab appears, he's full of multiple-entendres and self-references. He talks about himself, saying he "would crab up a storm." Of course, his name, the Crab, is self-reference within the dialog. But the Crab's words also hint at the structure of the dialogue. "'Which came first—the Crab, or the Gene?' That is to say, 'Which came last—the Gene, or the Crab?'" (His reference to his genes reflects on the structure of meaning in yet another way. Unless you're familiar with the genetics of crabs, you'll have to read *Godel, Escher, Bach* to appreciate the allusion.)

The conversation between the Tortoise and Achilles touches on a print by Escher in which interlocking crabs are formed by the figure and the ground, and also on a crab canon in Bach's *Musical Offering*. In less than four pages, the word "crab" has acquired a myriad of symbolic meanings.

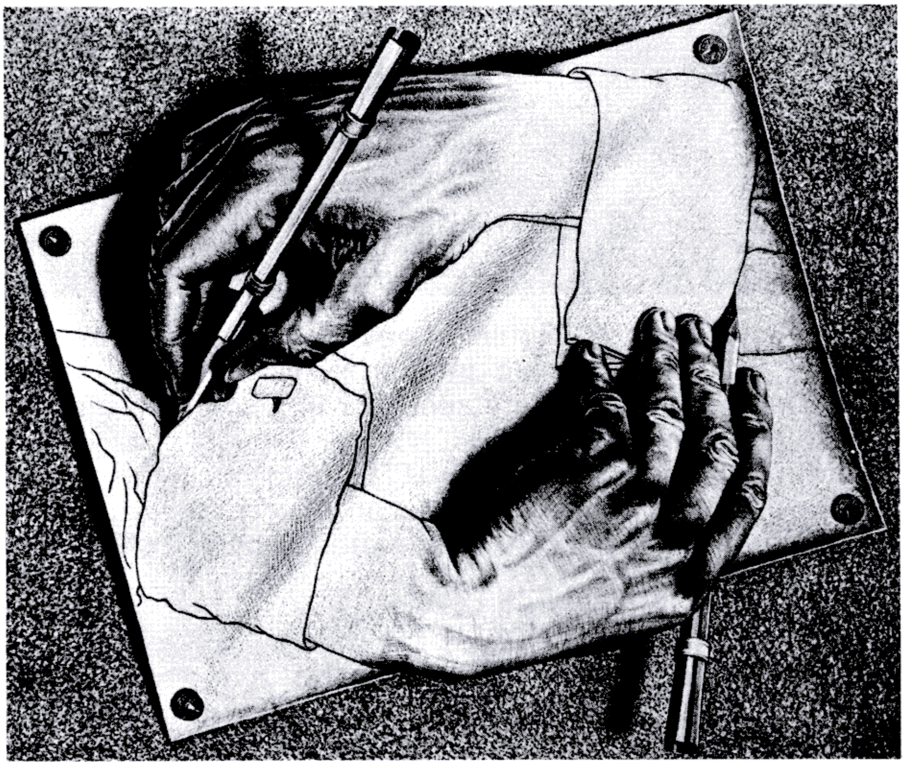
The "Crab Canon" is a highly structured piece of writing that causes numerous sets of operations to be carried out within the reader's cognitive system. But we use such deep and intertwined levels of interpretation daily. While I was reading *Godel, Escher, Bach*, I dreamed one night that I was an AI researcher. My dream is a less sophisticated story than Hofstader's "Crab Canon." But it is a clear example of several levels of implicit meaning within the hierarchy of my own thought processes.

An AI research project I had been working on for many years was reaching its culmination. Yet, I felt the answers to many of my questions were beyond my reach. I had access to a vast computer system and to several microcomputers. None of them could help me form ideas. I left the lab to walk along a secluded beach at twilight.

I walked up and down the beach for several hours. I contemplated subtle distinctions between the minds of men and the minds of machines. Finally, I laid down beneath a cluster of palm trees to sleep. As I listened to the waves, I thought to myself, "At least in this way, the human being will always be different from computers." I had struggled with ideas and depression, and I had been comforted by nature.

When I awoke at dawn one of my microcomputers was beside me. It presented me with an exhaustive catalog comparing and contrasting the most current facts about human and machine intelligence. I hadn't programmed the computer to generate the list. It had acted on its own volition.

In many ways, the dream is a synopsis of *Godel, Escher, Bach*. First, some thought patterns Hofstader stresses are evident. There's a circular pattern of self-reference,



M. C. Escher's Hand Drawing Hand

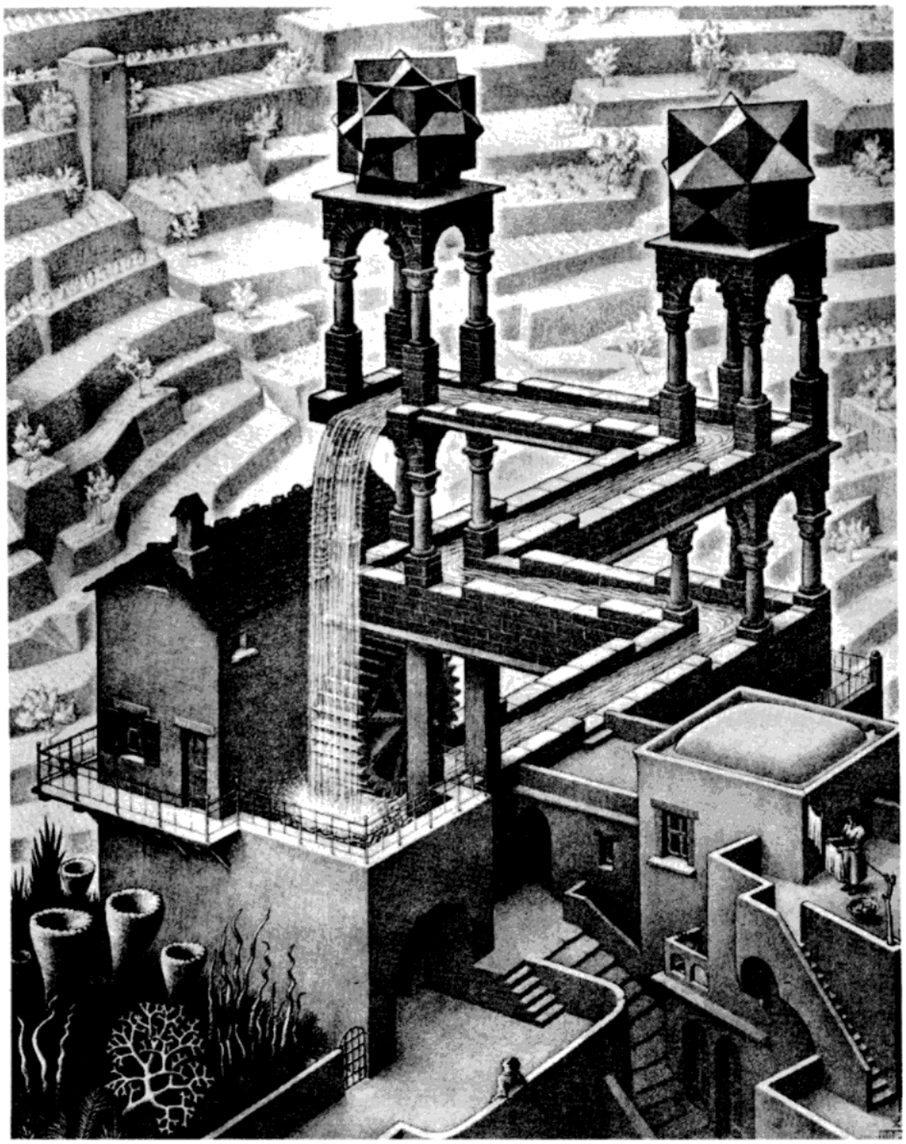
"The debate over artificial intelligence is a two-step strange loop, an infinite quest represented by an interplay between questions and answers."

not unlike a Strange Loop. While I was asleep I dreamed; and in that dream, I dreamed I fell asleep.

Within the dream, I awoke at dawn. For me the image of awaking at dawn is—through isomorphism—a symbol of human discovery. When I actually woke up the next morning, I'd gained insights to *Godel, Escher, Bach.*

The dream's synopsis of the book begins with its recognition that the distinctions between human and computer intelligence are apparent. Scientists and researchers have a relatively strong understanding of the thought processes of number computation, and to some extent of logic. These things we have programmed into computers with measured success.

The dream also recognizes that the gap between the two intelligences is currently the burden of human intelligence. To us volition and creativity are mysterious processes, and so far unprogrammable. The achievements of programs such as SHRDLU help us understand our own intelligence. And as our understanding of our own intricate thought processes increases, it seems inevitable that computer intelligence will advance. At the conclusion of the dream, the question "Can machines think?" is overshadowed by a new question. "Can machines feel emotion?"



M. C. Escher's Waterfall

Alan Turing, the computer scientist quoted at the opening of this article, believed the complexity of the operations and responses a machine is programmed to handle is related to the quality of its intelligence. Hofstader gives a great deal of consideration to Turing's suggested test for computer intelligence, and also to the common objections to machine intelligence which Turing anticipated. But Hofstader also considers a response to Turing's proposals by the philosopher J.R. Lucas. The following remarks are from Lucas' paper ***Minds, Machines, and Godel.*** A lengthy excerpt is included in *Godel, Escher, Bach.*

> "Complexity often does introduce qualitative differences. Although it sounds implausible, it might turn out that above a certain level of complexity, a machine ceased to be predictable, even in principle, and started doing things on its own account, or, to use a very revealing phrase, it might begin to have a mind of its own. It would begin to have a mind of its own when it was no longer entirely predictable and entirely docile, but was capable of doing things which we recognized as intelligent, and not just mistakes or random shots, but which we had not programmed into it. But then it would cease to be a machine, within the meaning of the act. What is at stake in the mechanist debate is not how minds are, or might be, brought into being, but how they operate."

Here is a new question, closely related to the question presented by the dream. If we learn how minds operate, and if a machine is programmed to act like a human mind, and if that machine acts of its own volition, is it still a machine?

A friend of Hofstader's once commented that AI should have its own incompleteness theorem because "Once some mental function is programmed, people soon cease to consider it an essential ingredient of 'real thinking.'" The paradox of artificial intelligence is that it exists (at some levels), but that existence cannot be proven. We know too little about human intelligence to bear the burden of proof.

The debate over artificial intelligence is a two-step strange loop, an infinite quest represented by an interplay between questions and answers. When one question is answered, another will always arise. This is the central message of *Godel, Escher, Bach: An Eternal Golden Braid.* ■

Nancy Robertson is the former news editor for 80 Microcomputing. *She is now living and writing in Peterborough.*

M. C. Escher photographs courtesy of the Vorpal Galleries.

✓See List of Advertisers on page 386

Correcting a failure to communicate.

Tape Regenerator

Cass R. Lewart
12 Georjean Dr.
Holmdel, NJ 07733

The tape recorder is an inexpensive and popular means of program and data storage for the TRS-80. Even disk owners still need the tape recorder to transfer purchased cassette programs to disk. Unfortunately the tape recorder brings its share of problems to the unwary. One of the most frustrating moments for a TRS-80 owner is when, after spending several minutes loading a new program, the letter "C" or a permanent double asterisk appears in the right upper corner of the monitor screen.

Frequently a Basic tape may seem to load correctly, but will produce a garbage listing.

Regardless of the symptoms, the diagnosis is the same—an irrecoverable error occurred during program reading. (At this point, grown men and women have been known to cry.) The only way to load a program from a poorly recorded tape is to find by repeated trial and error the very narrow range of volume control settings good for this particular tape. There is also no assurance that the volume control setting will work the next time around even with the same tape; a slight recorder head misadjustment may require repeating the whole procedure.

The Problems

The reason for difficulties in loading commercial tapes such as Radio Shack's is that they are reproduced on high-speed audio machines not specifically designed for digital encoding and not always kept in tip-top condition. The ideal signal wave form and the typical good and poor wave forms found on commercial copies of Level II programs are shown in Figs. 1 and 2. The superimposed noise, 60 Hz hum, amplitude distortion, ringing, and, what is worst—the incorrect displacement of the data pulses relative to the clock pulses—make it difficult or even impossible for the computer to read the tape.

If a single reading error occurs, e.g., one out of 100,000 pulses in a typical program is misread, then the whole program cannot be executed. Once a Basic program has been properly loaded one can make a good backup copy of it with the CSAVE command. It is not as straightforward with System programs. Monitors such as T-Bug, MON3, RSM or TRCopy will generate a backup copy of a System program, but with severe limitations, i.e., the original program has to be readable and the program cannot overwrite the monitor.

The Solution

The Tape Regenerator, on the other hand, will take any TRS-80 tape with Level II Basic or System programs, even if it is poorly recorded, and generate, using the TRS-80 computer, a backup copy of the programs on a second tape recorder. The Tape Regenerator circuit and the associated program Regen reshape and retime the pulses on the new tape, thus producing a tape that is easily read by the TRS-80 circuitry.

The Tape Regenerator does not care whether a program consists of multiple segments with separate loader or whether there is more than one program on a tape. Regen operates on one pair of clock/data pulses at a time, and unlike the monitor programs it does not store the whole program in computer memory. Thus tapes with multiple programs can be handled in a single run and even a 4K computer can back up copies of large programs.

It should be mentioned that several commercially available hardware devices (Data Dubbers) were designed for the same purpose as the Tape Regenerator.

T1	Audio transformer, R. Shack 273-1380 or equiv.
T2	12 Volt transformer, R. Shack 273-1385 or equiv.
D1,D2	Silicon diodes 50 Volt/1A
D3	3-6 Volt Zener diode
RECT	Bridge rectifier 50 Volt/1A
LED1,LED2,LED3	Red light emitting diodes
IC1	74LS02, Quad NOR gate
IC2	74LS367, Hex tri-state buffer
IC3	7805 Positive 5 Volt regulator
Q1	NPN silicon transistor, R. Shack 276-2014 or equiv.
R1,R3,R5	270 Ohm resistors
R2	10 Kohm resistor
R4,R6,R7	470 Ohm resistors
C1	200 MF/35 Volt capacitor
C2,C3	0.1 MF capacitors
S1,S2	SPDT switches
P1	2 by 20 pin edge connector, R. Shack 276-1558 or equiv.

A drilled glass epoxy silk screened printed circuit board, and a listing of Regen in Basic to allow POKEing it into memory can be purchased from C&R Electronics, P.O. Box 217, Holmdel, NJ 07733 for $10.95 ppd. New Jersey residents, please add 5% sales tax. Allow 2-3 weeks for delivery.

Table 1. Parts List

"The backup tapes produced from these devices. . . retain and may even worsen the timing jitter. . ."

However, these devices only reshape the clock and data pulses and do not retime them in respect to each other as Regen does. The backup tapes produced by these devices, though better in some respects than the originals (thanks to pulse reshaping) retain and may even worsen the timing jitter—one of the major causes of tape reading difficulties.

To test the soundness of the regeneration approach, multiple generations of the same program were made by generating tape 2 from 1, tape 3 from 2, etc. The test was run through five tape generations with a fairly long System program. It was found that a fifth generation tape would load on the computer as easily as the first generation tape with no apparent degradation in quality. Backup copies were also made of commercial tapes that, due to poor recording, would not load properly at dozens of different volume control settings. The backup copies loaded then with no difficulties.

Circuit Description

The circuit shown in Fig. 3 reshapes the clock and data pulses received from the first tape recorder and feeds them to the computer for processing. The audio transformer T1 provides DC separation between the tape recorder and the rest of the circuit. Switch S1 and diodes D1 and D2 allow selection of the better half of the pulse (see Control Adjustments). Zener diode D3 and transistor Q1 further shape the incoming signals.

The NOR gate in IC1 decodes the IN command from the computer indicating that it is ready to accept data for further processing. Spare gates in the IC1 serve as buffers and open the gate in IC2 to let the clock/data pulses pass via P1 connector to the data bus in the computer. LED1, LED3 and LED2 indicate respectively that the circuit is powered, that the Regen program is running and that the tape recorder is sending data at the proper level.

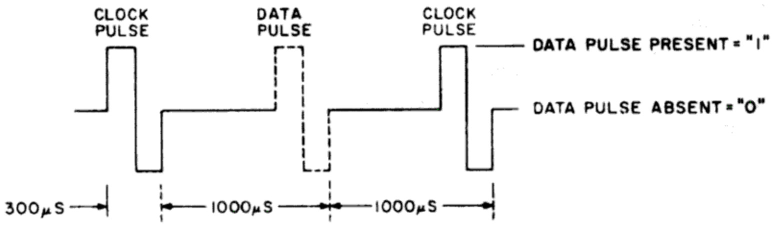


Fig. 1

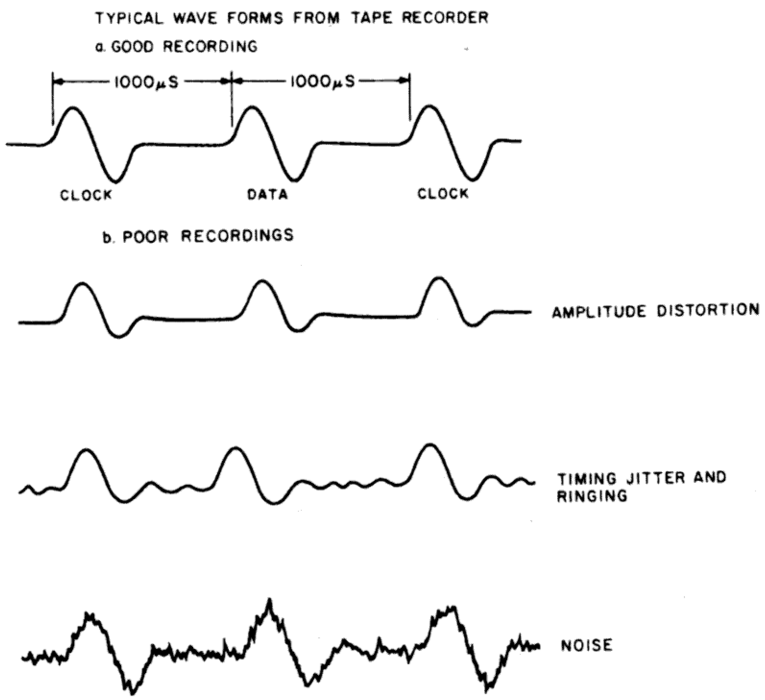
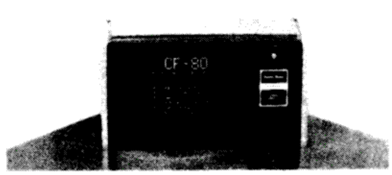


Fig. 2



See List of Advertisers on page 386

"The signal is partially reshaped... still it frequently can be read... where the original tape could not."

Transformer T2 in conjunction with the voltage regulator IC3, full wave rectifier RECT and two capacitors provides 5V to power the transistor and ICs. J2 provides a "quick and dirty" direct output to the TRS-80 if no backup copies are required. The signal at this point is partially reshaped but not retimed. Still it frequently can be read by the computer where the original tape could not be read.

Regenerator Program Regen

The Regen program shown in Listing 1 takes care of retiming and reshaping of Level II clock and data pulses. After clearing the screen, displaying a message with the author's name, and putting a graphic character in the upper right corner of the screen, the program waits for the clock pulse from the first tape recorder. After finding it another test is made to make sure that it was not a transient.

When the clock pulse is confirmed, it is put out after a 200-microsecond delay using subroutine output. This subroutine produces a perfect signal, as shown in Fig. 4, via the computer plug leading to the recording (AUX) jack on the second tape recorder. The graphic screen character is changed every time a clock pulse is detected to indicate that something is happening. A search for the data pulse now begins. A delay of 500 microseconds excludes any residual ringing from the preceding clock pulse.

If no data pulse is found during the following 700-microsecond window, the search for the next clock pulse starts again. If a data pulse is found, it is tested to exclude a transient; if the data pulse is confirmed, it is put out at the end of the 1000-microsecond interval that started at the beginning of the preceding clock pulse as shown in Fig. 4. A data pulse appearing any time between 500 microseconds and 1200 microseconds after a clock pulse is thus correctly retimed to occur exactly 1000 microseconds after the clock pulse.

The program then continues after a 100-microsecond delay with the search for the next clock pulse. The above-mentioned delay constants of course could be modified to operate on Level I programs, or to make Level II programs generated on the Model I computer more readable on the Model III computer, with its slightly different optimum tape-timing requirements.

Regen can be loaded either from Listing 1 using EDTASM or one of the monitor programs, or it can be loaded by running an equivalent Basic program mentioned in the parts list, which will POKE the program into memory.

Operating Instructions

1. Turn off the power, then make all connections between the tape recorders, computer, and the Tape Regenerator as shown in Fig. 5.

2. Turn the power on to the computer and the Tape Regenerator. LED1 should light up and the Memory Size question should appear on the screen. If it does not, check all connections, in particular the 40-pin connector from the Tape Regenerator to the CPU or the expansion interface. Repeat steps 1 and 2 if necessary.

3. Load the Regen program and run it. LED3 should now light up, author's name should appear in the middle of the screen, and a stationary graphic character should appear in the right upper corner. The program is in an infinite loop and will run until you reset the computer or turn it off.

4. Start playing the tape from tape recorder 1 and set the polarity switch and the volume control on the tape recorder as explained under Control Adjustments. Rewind the first tape recorder.

5. Start tape recorder 1 in play mode and tape recorder 2 in record mode. Observe LED2 and the changing graphic character on the screen. When LED2 goes dark and the graphic character stops changing, you are finished with the recording. Turn off

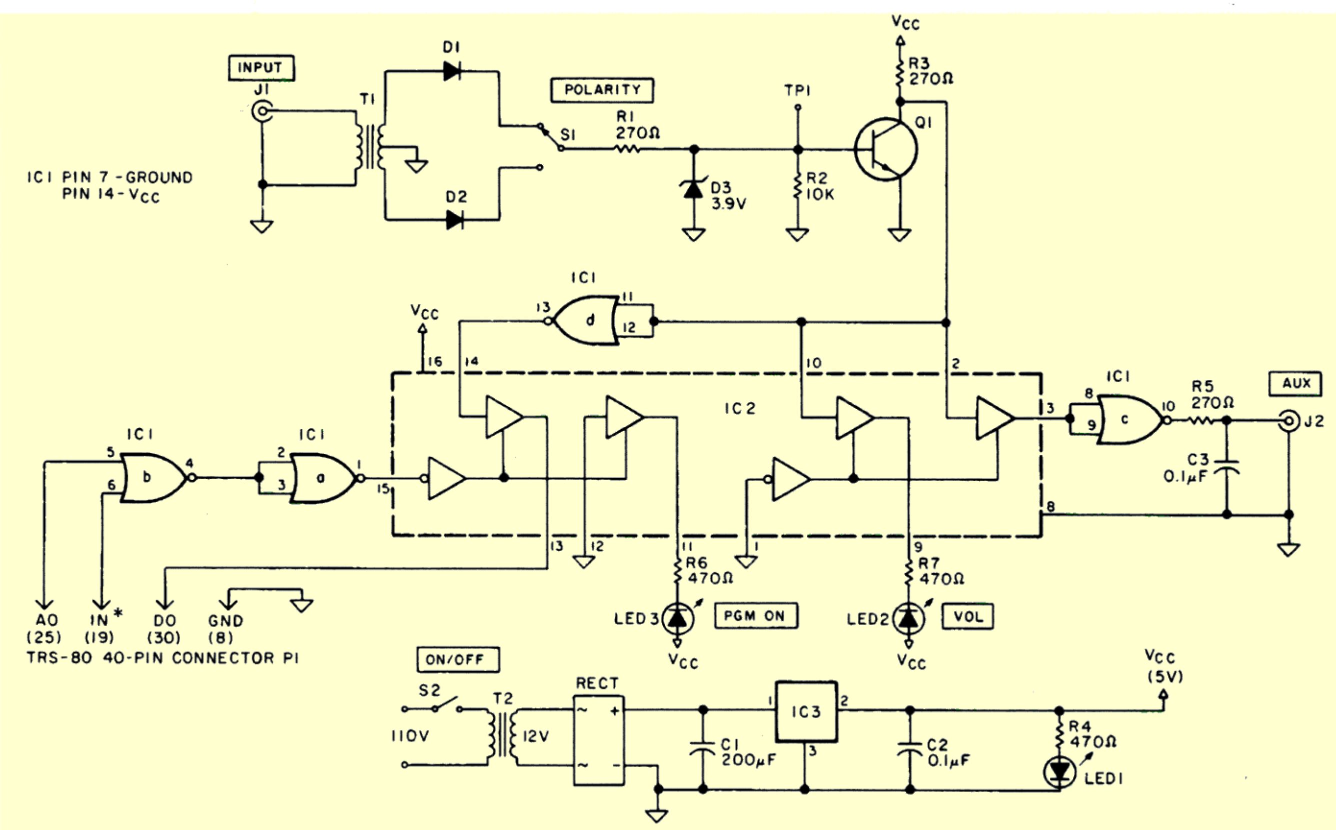
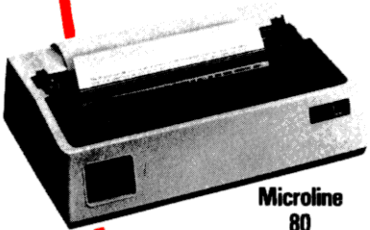


Fig. 3

This Months Special
Okidata Microline 80
At a New Low, Low, Price!



Microline 80

A great buy

– List Price is **$545.** The **Microline 80** is a quiet, small printer with field proven reliability.

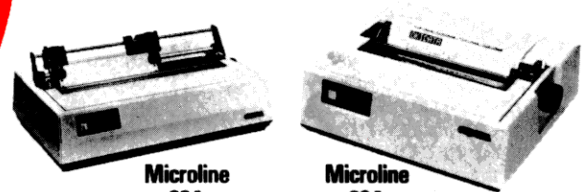
Features:

- Friction & pin paper feed, roll or fan fold
- Prints expanded characters and block graphics
- Print speed 80 characters per second
- Printhead rated at 200 million characters continuous cycle
- 132-column compressed print
- Available RS-232C Serial Interfaces:

	LIST
ORS-256 (256 Character Memory)	$109.00
MSP-100 (Unbuffered, Switch Select Data Rates)	$120.00
MSP-200X (200 Character Memory)	$215.00
MSP-2000X (2000 Character Memory)	$300.00
Other available options:	
Adjustable Tractor Feed - Model 54100201	$ 50.00
TRS-80 Parallel Output, cable to printer	$ 35.00
Ribbons	$ 3.00

TRANSFORM YOUR COMPUTER

into a Word Processing or Data Processing System by adding one of the NEW Okidata Printers



Microline 83A

Microline 82A

If you require a system printer, the new Okidata Microline 82A, 83A, 84 should be your choice.

These printers feature:

- 9 x 9 Dot Matrix heads rated at 200 million characters continuous duty - true lower descenders
- Full forms control – vertical tab, top of form
- Block Graphics
- Bidirectional, Short Line Seeking
- Dual Interface – Parallel & RS-232C Serial

	LIST
Microline 82A – 80-column,	
120 characters per second	$ 879.
Tractor Optional	$ 50.
Microline 83A – 136-column,	
120 cps, built-in tractor	$1195.
Microline 84 – 136-column,	
200 cps, built-in tractor	$1495.
Optional Serial Interfaces – RS-232C:	
HS-RS232 – 256 Character Buffer	$ 180.
HS-RS232 – 2K Character Buffer	$ 270.

THE STOCKING SOURCE ✓438

Visit our exhibit in Boston and Atlanta at the National Computer Show.

ASK FOR OUR INSTANT DISCOUNT From Roy Hawthorne Talk To Bill Tokar On Applications	CALL TOLL FREE U.S.A. 1-800-521-2764 MICHIGAN 1-800-482-8393	Remember: We are open 8:30 AM to 6:00 PM EST Monday – Friday	master charge VISA No Credit Card Penalty	"The Stocking Source" 24069 Research Drive Farmington Hills, MI 48024 313-474-6708

✓See List of Advertisers on page 386

"The Tape Regenerator will only process tapes having complete program information."

everything. The 40-pin connector can be left in place as it does not interfere with normal computer operation.

Control Adjustments

There are two controls to be set to properly read a poorly recorded tape: the polarity switch in the regenerator circuit and the volume control on the tape recorder. Make both settings while the tape recorder 1 is in play mode. The settings will vary from tape to tape depending on the machine on which it was recorded. However, your own backup tapes produced with the Tape Regenerator or CSAVE command should all work with the same settings. First, set the polarity switch to the position giving a stronger signal as evidenced by LED2 being brighter at a low volume control setting. Then find the optimum volume control setting by one of the following methods:

- Turn up the volume control until LED2 lights up brightly, then back up slightly.
- Turn the volume control slightly above the point when the graphic character in the corner of the screen starts changing.
- Observe the incoming pulses with a scope at point TP1 (best method) and adjust the volume control for cleanest and widest pulse shape.

Program Limitations

The Tape Generator will only process tapes having complete program information. Tapes with eighteen-minute gaps (see history books) or those with pulses of very unsteady amplitude or embedded in noise are not acceptable. Neither the Tape Regenerator nor any other device will restore

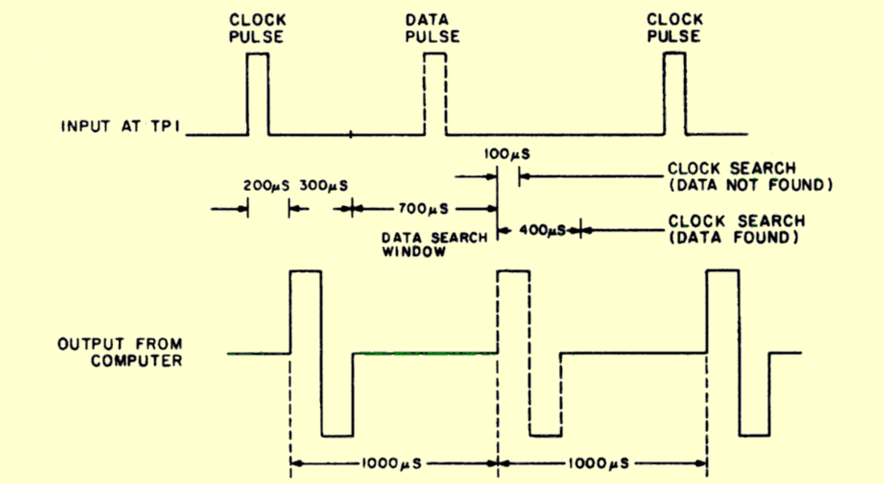


Fig. 4

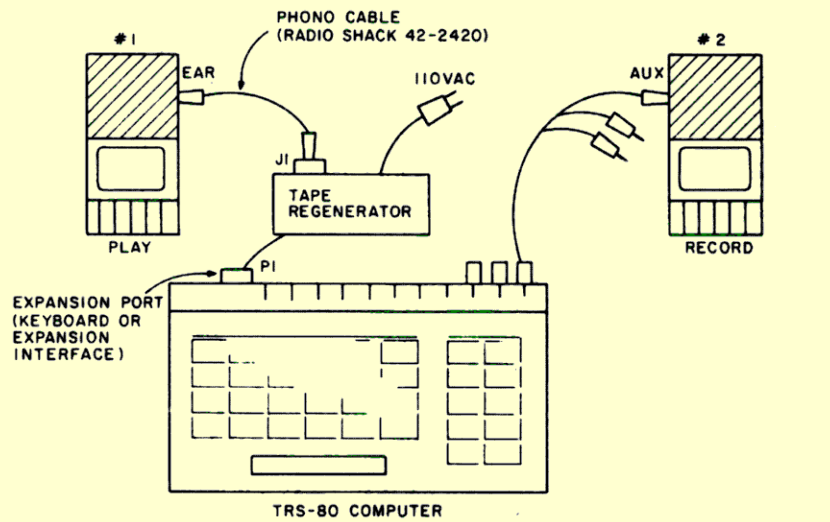


Fig. 5

them. The hardware and software described here will only work with a Model I, Level II computer.

Construction Hints

Keep all wires as short as possible; use of a printed circuit board is highly recommended. As only four contacts are being used on the 40-pin connector you can spread the remaining contacts with a screwdriver for easy insertion. Before turning the power on, check all connections. Proper care may prevent unexpected fireworks.

"Neither the Tape Regenerator nor any other device will restore them."

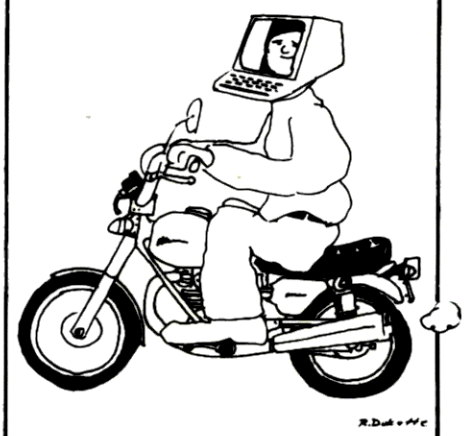
Note for Disk Users

The Regen program is fully relocatable and can be stored at any convenient location in memory not interfering with DOS. The DI instruction at the beginning of the program will assure that the internal clock, which would interfere with tape operation, is not running. ■

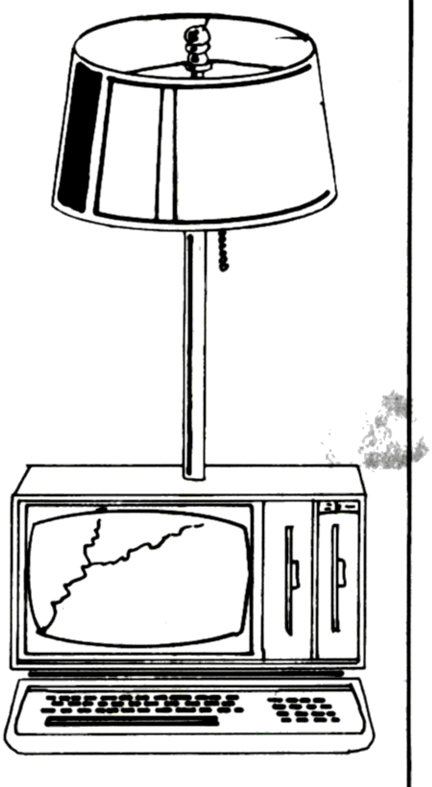
```
0000                PORT     EQU     00H
0004                CENTER   EQU     04H
0005                HIGH     EQU     05H
0006                LOW      EQU     06H
0009                DL100    EQU     09
0012                DL150    EQU     18
0014                LEN1     EQU     20
001A                DL200    EQU     26
0022                DL700    EQU     34
0022                LEN2     EQU     34
00FF                CASS     EQU     0FFH
01C9                CLS      EQU     01C9H
3C3F                BLINK    EQU     3C3FH
3DD6                LINE1    EQU     3DD6H
3E0F                LINE2    EQU     3E0FH
4A00                         ORG     4A00H
4A00 CDC901         REGEN    CALL    CLS             ;CLEAR SCREEN
4A03 F3                      DI                      ;DISABLE INTERNAL CLOCK
4A04 216D4A                  LD      HL,TAB1         ;DISPLAY LINE1
4A07 11D63D                  LD      DE,LINE1
4A0A 011400                  LD      BC,LEN1
4A0D EDB0                    LDIR
4A0F 21814A                  LD      HL,TAB2         ;DISPLAY LINE2
4A12 110F3E                  LD      DE,LINE2
4A15 012200                  LD      BC,LEN2
4A18 EDB0                    LDIR
4A1A 213F3C                  LD      HL,BLINK        ;POINTS TO SCREEN CORNER
4A1D 0609           START    LD      B,DL100         ;100 MICS DELAY
4A1F 10FE           DEL0     DJNZ    DEL0
4A21 7E                      LD      A,(HL)
4A22 3C                      INC     A               ;CHANGE GRAPHIC BLOCK
4A23 F680                    OR      80H
4A25 E6BF                    AND     0BFH            ;ASSURE GRAPHIC CHARACTER
4A27 77                      LD      (HL),A          ;DISPLAY NEW CHARACTER
4A28 DB00           SRCHC    IN      A,(PORT)        ;START CLOCK PULSE SEARCH
4A2A 1F                      RRA
4A2B 30FB                    JR      NC,SRCHC        ;CLOCK PULSE FOUND?
4A2D DB00                    IN      A,(PORT)        ;YES, TRANSIENT ONLY?
4A2F 1F                      RRA
4A30 30F6                    JR      NC,SRCHC        ;YES, KEEP SEARCHING
4A32 061A                    LD      B,DL200         ;NO, SET 200 MICS DELAY
4A34 10FE           DEL1     DJNZ    DEL1
4A36 CD584A                  CALL    OUTPUT          ;PUT OUT CLOCK PULSE
4A39 0622                    LD      B,DL700         ;START 700 MICS READ WINDOW
4A3B DB00           SRCHD    IN      A,(PORT)        ;SEARCH FOR DATA PULSE
4A3D 1F                      RRA
4A3E 3804                    JR      C,FOUND1        ;FOUND?
4A40 10F9                    DJNZ    SRCHD           ;NO, WINDOW TIMED OUT?
4A42 18D9                    JR      START           ;YES, SEARCH FOR CLOCK PULSE
4A44 DB00           FOUND1   IN      A,(PORT)        ;TRANSIENT ONLY?
4A46 1F                      RRA
4A47 3804                    JR      C,FOUND2        ;NO
4A49 10F0                    DJNZ    SRCHD           ;YES, WINDOW TIMED OUT?
4A4B 18D0                    JR      START           ;YES, SEARCH FOR CLOCK PULSE
                                                     ;DATA PULSE FOUND,
4A4D DD23           FOUND2   INC     IX              ;WASTE 10 CYCLES
4A4F CB5E                    BIT     3,(HL)          ;WASTE 12 CYCLES
4A51 10FA                    DJNZ    FOUND2          ;WINDOW TIMED OUT?
4A53 CD584A                  CALL    OUTPUT          ;YES, PUT OUT DATA PULSE
4A56 18C5                    JR      START           ;SEARCH FOR CLOCK PULSE
4A58 3E05           OUTPUT   LD      A,HIGH          ;PULSE OUTPUT
4A5A D3FF                    OUT     (CASS),A        ;PULSE HIGH
4A5C 0612                    LD      B,DL150         ;150 MICS DELAY
4A5E 10FE           DEL2     DJNZ    DEL2
4A60 3E06                    LD      A,LOW
4A62 D3FF                    OUT     (CASS),A        ;PULSE LOW
4A64 0612                    LD      B,DL150         ;150 MICS DELAY
4A66 10FE           DEL3     DJNZ    DEL3
4A68 3E04                    LD      A,CENTER
4A6A D3FF                    OUT     (CASS),A        ;RESTORE TO CENTER
4A6C C9                      RET
4A6D 54             TAB1     DEFM    'TAPE REGEN   PROGRAM'
4A81 43             TAB2     DEFM    'COPYRIGHT (C) 1981  CASS R. LEWART'
4A00                         END     REGEN
00000 TOTAL ERRORS
```

Listing 1. Regen Program

101 USES FOR A DEAD MICROCOMPUTER no.45



101 USES FOR A DEAD MICROCOMPUTER no.5



✔See List of Advertisers on page 386

Make your computer a sophisticated 80.

Real World Interface—Part I

Elliott K. Rand
P.O. Box 552
Melbourne, FL 32901

With a few trivial exceptions, every byte of information in your system got there through a keyboard. Missing is a real-time, *now* communication with the *physical* world: complex communication, not just turning a switch on or off. Sensing the real world, the computer can be programmed to respond intelligently and control the outside world. With additional sensors to monitor its action, the system becomes self-adjusting.

The world is continuous—temperatures do not jump suddenly from 50 degrees to 70 degrees; velocities do not change from 0 to 60 mph in an instant. However, computers use numbers that appear to change abruptly. There is no such thing as a fraction of a bit. Obviously, some translation is needed in both directions.

Input/Output

Our goal is to get information on the data

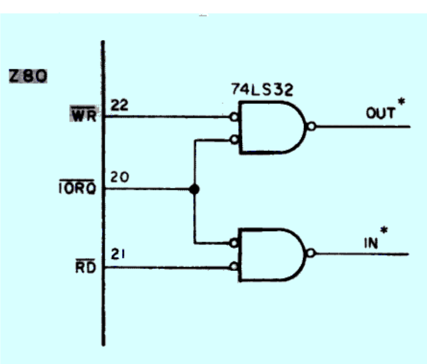


Fig. 1. I/O Control = Bus Signals in the TRS-80.

A←(n)

Data byte at Port (n) is transferred to accumulator.

Format:

Opcode	Operands
IN	A,(n)

1	1	0	1	1	0	1	1	DBH

← n →

Description:

The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator also appear on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into the Accumulator (register A) in the CPU.

M CYCLES: 3 T STATES: 11(4,3,4) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H and the byte 7BH is available at the peripheral device mapped to I/O port address O1H, then after the execution of

IN A,(01H)

the Accumulator will contain 7BH.

Fig. 2. Out Control = Bus Instruction.

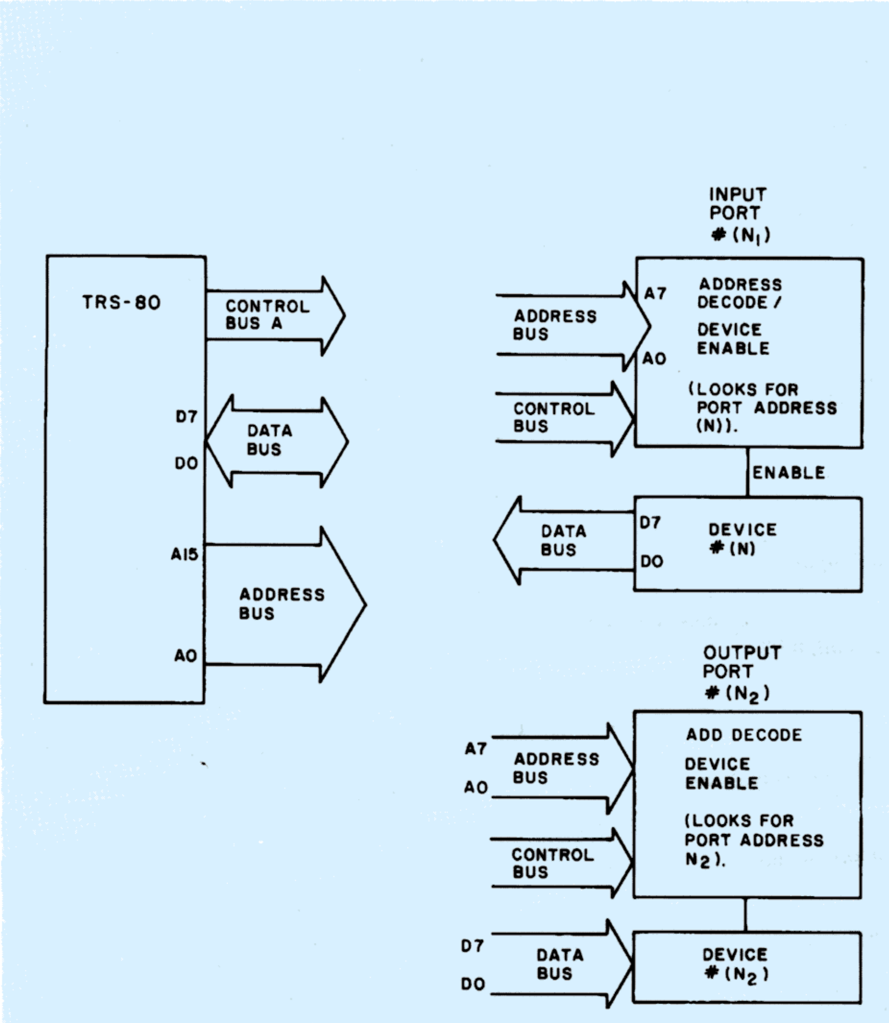


Fig. 3. Connection of data bus to data register of external device.

Operation: A←(n)

Format:

Opcode	Operands
OUT	(n),A

1	1	0	1	0	0	1	1	D3H

← n →

Description:

The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator also (register A) appear on the top half (A8 through A15) of the address bus at this time. Then the byte contained in the Accumulator is placed on the data bus and written into the selected peripheral device.

M CYCLES: 3 T STATES: 11(4,3,4) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H, then after the execution of

OUT (01H),A

the byte 23H will have been written to the peripheral device mapped to I/O port address 01H.

Fig. 4. In Control = Bus Instruction.

✓See List of Advertisers on page 386

"Apply power to the real world interface; all LED's should be lit."

bus into the outside world and convert information from that world into a format acceptable to the data bus. This information must arrive at the data bus at a precise time; otherwise it will be missed or conflict with other signals, resulting in chaos. Properly synchronized signals are generated by the central processing unit (CPU) for I/O control.

Fig. 1 shows how In and Out control-bus signals are generated. The Z-80 responds to "OUT (N), A" (Fig. 2), by outputting a logic 0 on pins 20 (IORQ*) and 22 (WR*). A logic 1 appears on pin 21 (RD*) as the Z-80 forbids RD* and WR* signals from being low simultaneously. The IORQ* and WR* signals are AND-ed in a 74LS32 and buffered by a 74LS367 providing the TRS-80 OUT* control-bus signal whenever those outputs are low, as indicated by the asterisks.

At the same time, the CPU outputs the second byte of the instruction (a binary number between 00000000 and 11111111) onto the lower eight lines of the address bus causing connection of the data bus to the data register of the external device (see Fig. 3).

The "OUT (N), A" instruction also transfers data from the Z-80 accumulator to a latched data register in the external device.

The "IN A, (N)" command (Fig. 4) is similarly handled. Data flow is from the external device to the Z-80 accumulator, Register A. Input normally does not require latching hardware.

Instead of the variety of integrated circuits required for signal control, output latching and input buffering, an 8255 Programmable Peripheral Interface is used. Depending on software, this 40-pin device has three eight-bit ports which may be used for input or output; a tri-state bi-directional bus transceiver; or a 12-bit output port with an eight-bit ≠ input port and a four-bit status port.

For our purposes the 8255 is programmed for simple I/O with ports A and B as latched output ports, and port C as an unlatched input port. Port B is uncommitted.

Some Introductory Experiments

Fig. 5. shows the real world interface circuitry required for these experiments. Integrated circuits U5, U6, U7 and transistors Q1 and Q2 are not used and need not be installed. A well-regulated (4.75–5.25 v. max.) +5-volt DC supply capable of delivering one ampere is required (see Fig. 6).

Before attempting these experiments remove the power from your TRS-80 and the power supply. *Never* leave power on while setting up test hardware.

Before installing the interconnect cable to the TRS-80 and the real world interface, connect the power supply to the real world interface's +5 and ground terminals. Turn on the power supply. Eight light-emitting diodes should light up. If not, disconnect the power supply and examine the board for soldering defects.

If all indications are normal, turn off the power supply, but do not disconnect it.

Install the 40-pin card-edge end of the interconnect cable to the TRS-80 expansion port with the ribbon cable exiting upward. Pin 1 of the 40-pin DIP plug goes to pin 1 of socket SO 1.

Apply power to the real world interface;

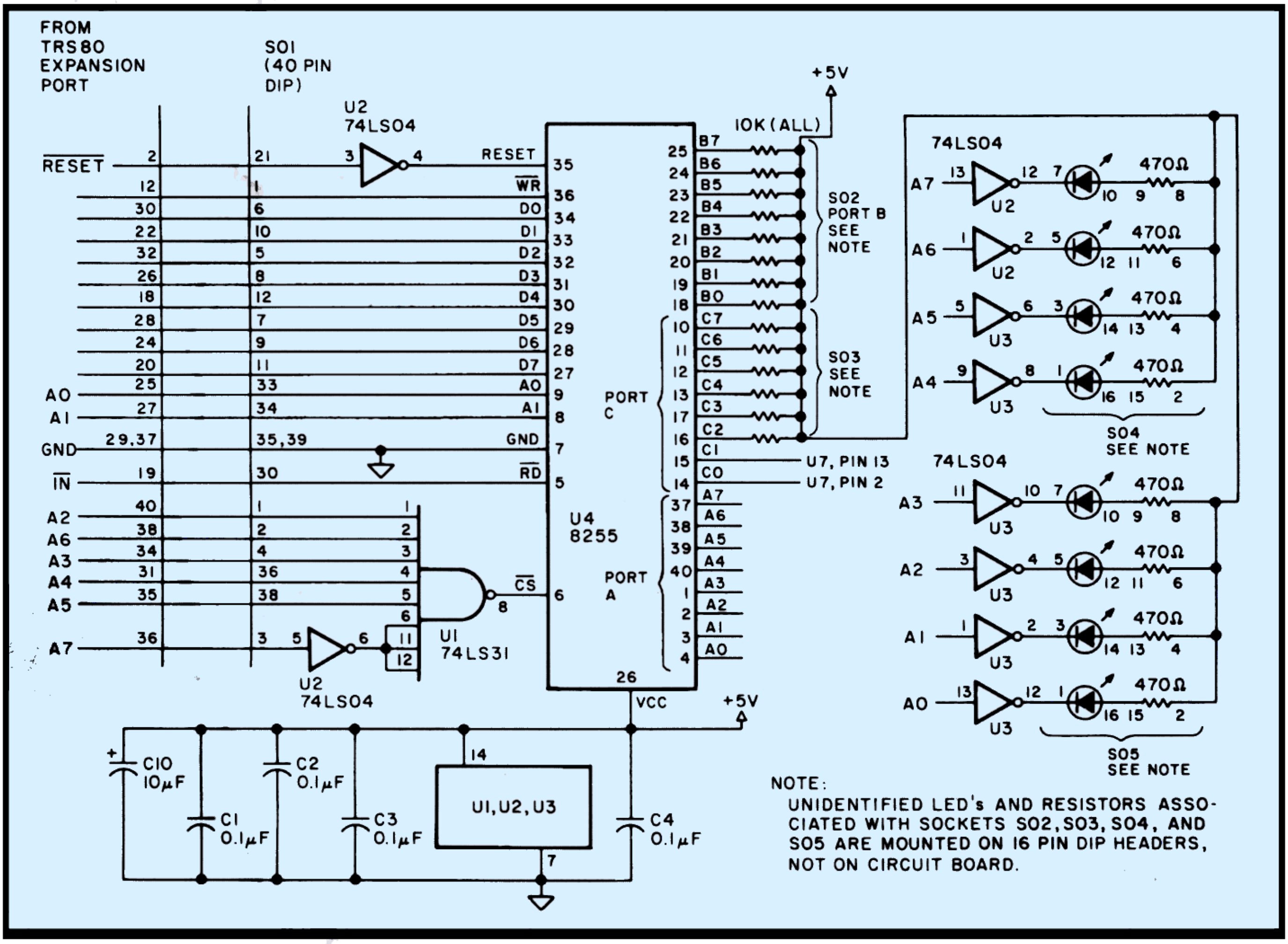
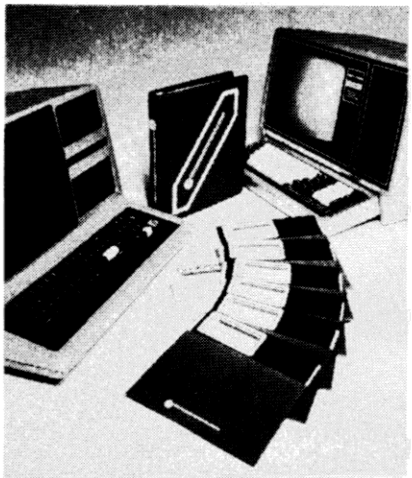


Fig. 5. Real-World Interface Circuitry.

ANNOUNCING A REVOLUTION IN THE COST OF PROFESSIONAL SOFTWARE



VISACCOUNT is a fully integrated business and accounting system designed for use in small businesses. VisAccount is extremely comprehensive and professional, yet it is very easy to use. The system is controlled from a series of interconnected menus permitting user-friendly operation. Everything you need to set-up and operate the system is provided with the VisAccount package.

ALL SYSTEMS is able to provide you VisAccount at this incredible low price because it runs on so many different machines. Through volume sales we are able to substantially reduce our prices.

OUR GUARANTEE — Buy both our software and that of our competitors (who will no doubt charge several times our price because they need to recapture their development cost). Compare the two systems and we know you'll return theirs (make sure they'll let you return their software). If you decide not to keep our system, then return it within 45 days for a full refund. Once you've used our system we're confident you'll be delighted.

† Microcomputers for Business Applications, 1979

VISACCOUNT™

What You Receive

- Nine 5¼" double density disks (or six 8" single density disks)
- Easy-to-use operator's manual (over 200 pages)
- Self-study guide on bookkeeping and accounting (over 180 pages)
- Cassette based instruction program on set-up and operation

Available for TRS-80 I, II & III, Apple* and most others.
*The Apple version requires the Microsoft Z80 softcard.
CSCA has CBASIC2, CP/M and Microsoft Z80 softcard in stock.

FREE: MAILING LIST PROGRAM
Requirements: 48K CBASIC 2
2 DISK DRIVES CP/M

ALSO

WORD STAR	$290
SUPERCALC	$249
DATA BASE SYSTEM	$190
CP/M for TRS80 MODEL II	$ 99
TERMINAL PROGRAM for MODEL II CP/M	$ 39

Send $159 for the VISACCOUNT system

ALL SYSTEMS Computer Services Corporation of America ✓59
332 East 30th Street New York, New York 10016
Order Toll Free **1-800-221-2486**
For New York and Technical Number 1-212-685-0090

Name ______________________

Address ______________________

City/State/Zip ______________________

☐ Master Charge ☐ Visa ☐ American Express

No. ______________ Expires ______________

Your System ______________________

Disk Size ☐ 5¼" double density ☐ 8" single density

"You may return to Basic at any time without losing programs in memory."

all LED's should be lit.

Turn on the TRS-80 power. A normal memory size start-up routine should occur. No memory need be reserved for these experiments, so press Enter to get into Level II Basic.

Enter and run the following Basic program:

```
10 OUT 127, 137     ('Configures 8255)
20 OUT 124, 85      ('Loads 55H into Port A)
```

Every second LED should be lit. Change line 20 to:

```
20 OUT 124, 170    ('Loads AAH into Port A)
```

Run it. You should observe the first, third, fifth and seventh LED lit. You may want to try other values in the second term of line 20.

Next enter and run the following Basic program:

```
10 OUT 127, 137
20 FOR N=0 to 255
30 OUT 124, N:NEXT N
40 GOTO 20
```

The LEDs should continuously cycle from zero through 255. Because the count is rapid, the two least significant digits on the right appear to be on all the time.

Insert the following line into the program and run it:

```
25 FOR M=0 TO 100:NEXT M
```

Now the two least significant (right-hand) digits should clearly blink on and off and the count can be visually followed.

Next, enter the same program in machine code using either the Basic POKE command, or load the T-Bug, RSM or EDTASM System programs. Keying in the hex words with T-Bug or RSM is fastest and easiest (see Table 1).

Refer to your system's instructions in loading this program. The first few instructions using the POKE command appear as follows:

```
10 POKE 24576,62:POKE 24577,137:POKE24578,211:
   POKE 24579,127 (etc.)
```

Decimal Address	Decimal Instruction	Hex Address	Hex Instruction	Mnemonic
24576	62	6000	3E 89	LD A, 89H
24577	137			
24578	211	6002	D3 7F	OUT 7F
24579	127			
24580	22	6004	16 00	LD D, 00H
24581	00			
24582	122	6006	7A	LD A, D
24583	211	6007	D3 7C	OUT 7C,A
24584	124			
24585	20	6009	14	INC D
24586	58	600A	3A 40 38	LD A, (3840H)
24587	64			
24588	56			
24589	230	600D	E6 04	AND 04
24590	04			
24591	194	600F	C2 19 1A	JP NZ,1A19H
24592	25			
24593	26			
24594	24	6012	18 F2	JR 6006H
24595	242			

Table 1.

Execute the program at 6000H. In Basic this is done by entering the System command. When an asterisk appears on your screen, type /24576 and press Enter to commence execution at 6000H. All the LEDs should be on, although the more significant (left-hand) bits may appear slightly dimmer than the least significant bits. The microprocessor is counting near its maximum speed—around 100,000 counts per second.

Next, delete the last line of the program and insert Table 2.

Execute the program beginning at 6000H. The instructions slow the count by a factor of approximately 300. The count now appears at about the same rate as the Basic program running at its fastest speed.

To slow the machine code program down to where all the bits may be visually observed, delete the last three program instructions and insert Table 3.

These steps slow the microprocessor by a factor of around 100,000. You may want to try loading different values into register pair BC. Loading BC with 0000H decrements it 2^{16} (65,536) counts before both B and C are zero again. Loading BC with 0001H causes the minimum delay.

You may return to Basic at any time without going through Reset and losing programs in memory by pressing and holding the Break key until the Basic prompt appears. It may take a second to work through the delay routine and get to your break signal. You may return to the machine code routine by exiting Basic with a System command, then answer the asterisk prompt as before.

When in Basic, enter and run:

```
10   OUT 127, 137
20   CLS:INPUT "ENTER A NUMBER BETWEEN 0 AND
     255 INCLUSIVE";N
25   IF N<0 OR N>255 GOTO 20
30   OUT 124,N
35   S$=INKEY $: REM WAIT FOR ANY KEY DOWN
40   GOTO 20
```

For each decimal number entered the LEDs should indicate the equivalent binary number.

Here's one for the Trekkies. Enter and run:

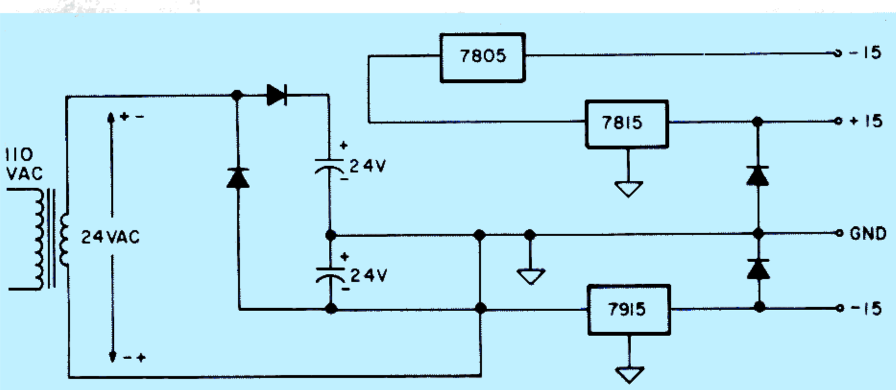


Fig. 6. Typical +5 volt, and ±15 volt direct current supply.

"With a few trivial exceptions, every byte in your system got there through a keyboard."

```
5     OUT 127, 137
6     CLS: PRINT@320 "PRESS 'L' TO GO LEFT.
      PRESS 'R' TO GO RIGHT
10    A$ = INKEY$:IF A$ = "L" GOTO 30
20    IF A$ = "R" GOTO 80
25    GOTO 10
30    A = 1
40    OUT 124, A
45    FOR X = 1 TO 10:NEXT X
50    C$ = INKEY$:IF C$ = "R" GOTO 80
60    A = 2*A: IF A>129 GOTO 30
70    GOTO 40
80    A = 128
90    OUT 124, A
95    FOR X = 1 TO 10:NEXT X
100   C$ = INKEY$:IF C$ = "L" GOTO 30
110   A = A/2
120   IF A<1 GOTO 80
130   GOTO 90
```

Press L to go left. Press R to go right. Neat. This is still just a fancy version of turning a switch on and off.

Now that digital information is released from the confines of the personal computer, it remains to be converted into a form meaningful to the physical world.

Next month we will explore the applications of digital-to-analog conversion as a means of sophisticated control. ■

24594	06	6012	06 00	LD B, 0
24595	00			
24596	16	6014	10 FE	DJNZ FE
24597	254			
24598	24	6016	18 EE	JR 6006
24599	238			

Table 2.

24594	01	6012	01 00 00	LD BC, 00 00H
24595	00			
24596	00	6015	0B	DEC BC
24597	11			
24598	121	6016	79	LD A, C
24599	176	6017	B0	OR B
24600	32	6018	20 FB	JR FB
24601	251			
24602	24	601A	18 EA	JR 6006
24603	234			

Table 3.

The greatest program Radio Shack has released.

Profile

Profile
Tandy/Radio Shack
Ft. Worth, TX
$79.95 Models I & III
$180 Model II

Bill O'Brien
11 Dongan Place
New York, NY 10040

Profile is probably the greatest program Radio Shack has released, far surpassing the potential of Microfiles, although it costs $20 less.

What makes it so great? The higher priced Microfiles lack useability; the user can access data stored in Profile, using it to provide information for programs which can be manipulated and output outside the Profile environment.

Profile is a data storage/retrieval system available for the TRS-80 Model I. It consists of four sub-programs—Init, Access, Print and Profile—and uses Disk Basic with one drive, with additional drives and a line printer optional.

The User's Manual begins with a general description of the terms used throughout the instructions, and the reminder that you cannot use the lowercase driver to enter data. Standard computer format is data entry with uppercase characters only. Lowercase is great for word processing, but it's not that great a loss. (For those of you keeping score, Microfiles has the same restriction.)

Profile is a machine language program that loads through DOS and asks the user, "How Many Drives (1–4)?" Be sure you have the program disk in drive 0, there are calls to other routines stored on that disk. Also, a formatted disk should be in any other drives.

In a few seconds your screen will be ablaze with life as the Profile program takes control. You will see a program header at the top of your screen, what appears to be operating instructions at the bottom of the screen (they are non-functional at this point in the program), and below that, the important message: Set Up Your Form—Press <Break> When Done.

Creating the Field

First, using the arrow keys to move the cursor, retitle the file by overwriting the program title line: **PROFILE DATA FORM** can be changed to: **MY DATA FILE**, or any other heading you want to give it. Then create the fields you want to store data in:

```
ACCOUNT NAME: .......................
    INVOICE#: ......
        DATE: ../../..
      AMOUNT: .......
     PO/JOB#: .......
 POSTED/PAID: ../../..
    CHECKED#: ....
```

Field headings can be whatever you choose, and are created by typing each literal (the field names are literals, while the values we will insert are variables), a colon and a space (which Profile recognizes as a delimiter), and whatever characters you decide to use to mark the size of the fields. (Periods are acceptable, but, as in the example, other characters are permissable—dollar signs, for instance, to mark monetary fields.) Keep in mind that the top line of the screen is used solely for the file title, and the total space available for field information is 255 characters. If you exceed this limitation (after entering all those dots and slashes) a Data Too Long message appears and you must reformat the file content.

If you find that you've made a mistake anywhere, use the arrow keys to position the cursor over the error, or use <shift> D to delete a character, or <shift> I to insert a space (to add a character or align the field names for a professional look). Make sure that once you finish, the field names and lengths are exactly the way you want them. When you press the Break key you're stuck with it, and making changes later is a lot of trouble.

Sorting

After entering data you can sort it by any of the fields you established. When the program looks at an entry, it sees it exactly as it has been typed. If we sort by a numeric field in ascending order (included in Profile is a search routine where the same rules apply), and two of the entries are $1104.20 and $9.68, something strange will happen.

Any common fool can tell that $1100 is more than $9, but the computer is ~~no~~ common fool; it's a very special kind of half-wit that takes everything literally. Therefore, as the computer sees it, nine is greater than one, so one comes first. Any numeric entry should have all spaces filled with zeroes. If our field is seven characters long, then 0023.89, 1404.13, and 0002.66 will produce the correct sort and search responses.

Searching for a particular entry or group of entries is accomplished by stating the field heading, and then specifying what qualities the found entries should have in relation to a comparator, which you enter using FORTRAN-like specifiers (NE, EQ, GT, LT, LE or GE for not equal, equal, greater than, less than, less than or equal to, and greater than or equal to.) For example, if you have an invoice number you want to search for, use INVOICE# as the field, EQ as the comparator and the number in response to the Search For question. If you want a range over or above the number specified, use GT and LT, or GE and LE if you want the entered number included.

The final option available is the printout.

See List of Advertisers on page 386

". . . the computer is no common fool; it's a very special kind of half-wit that takes everything literally."

During the display/edit mode you can get hard copy of the entry displayed.

If it is your first run through the printing section, you will be asked to construct a print format. This consists of three lines of data: the title line, the column headings and the data line(s). One hundred and thirty-two columns of print are available; the screen scrolls sideways as you pass the 64-column mark that would normally terminate a video line. Again, the choice of literals is up to you, although title headings should have something to do with the field headings you entered. The field headings themselves may be used as column titles.

When you're done, tell the computer which fields are to be placed in what spaces. You then specify which field you want, its relationship to the comparator you've selected, and the program begins to print.

Fooling Profile

The one thing Microfiles has over Profile is that it will work with a serial printer whose driver you have loaded into memory while Profile won't. Whoever created the program included a routine to check if there was a printer attached to the parallel port (there's a memory location in the machine that gets loaded with a certain value; if that value isn't there, Printer Not Ready is written on the screen).

Remember I said it was a machine language program: and you're as big a klutz with machine language as I am, right? Don't fear, pilgrim, there is something you can do (no, not call Texas). Have you got a copy of NEWDOS or T-BUG around? (Aw, c'mon, *everybody's* got a copy of T-BUG.)

The offending portion of the code in the Print program looks like this:

```
7C84 3AE837   LD     A,(37E8H)
7C87 FE80     CP     80H
7C89 381D     JR     C,7CA8H
7C8B 21003E   LD     HL,3E00H
7C8E 222040   LD     (4020H),HL
7C91 214875   LD     HL,7548H
7C94 CD2752   CALL   5227H
7C97 CD4552   CALL   5245H
7C9A CDCE76   CALL   76CEH
7C9D FE01     CP     01H
7C9F CA2152   JP     Z,5221H
7CA2 FE0D     CP     0DH
7CA4 28DE     JR     Z,7C84H
7CA6 18EF     JR     7C97H
```

Bear with me, its not that complicated. I did it right, and as far as I'm concerned, Bill Barden might as well be reciting *The Epic of Gilgamesh* in the original Sumerian.

The NEWDOS Fix

If you have NEWDOS, run Superzap. With a backup copy of Profile in the drive, select the DD option, the disk drive number, track C, sector 9 (that's the position it was in on my copy). Look for the sector display that matches Fig. 1. Type MOD C1. (This is the position in the sector of the first disk location we will modify.) The screen will blink, an M will appear at the beginning of the line and you can now proceed to modify it. To do this, type "0" 72 times, covering the values 3A through EF. Press Enter and Y, making the modification permanent. This changes the instructions in the listing above to NOP, which is the same in machine language as a line number, and a colon followed by a blank line in Basic. The computer knows the lines are there, but skips over them, since there is No OPeration to be performed.

We just changed the code that told the computer what it is supposed to do. Originally it was supposed to load a value into the A register from memory location 37E8(hex) (LD is a machine code load instruction), compare it (CP is machine language for compare) to the hex value 80H (128 decimal), and if it found them to be the same, proceed to memory location 7CA8(hex) (JR C,7CA8H is a jump instruc-

```
10C900   7221  BB71  0684  7EFE  2020  1A2B  1001  82D0  .!.........+....
10C910   7BF8  2100  3E22  2040  2107  75CD  3952  0602  ..!.>".@!...9R..
10C920   CD3C  52CD  3F77  C39A  77AF  32BD  7206  84DD  .<R.?.....2.....
10C930   21BF  72FD  2121  70FD  3600  0121  3872  7EFE  !...!!..6..!8...
10C940   2028  2DCD  C57A  0E01  FD34  0023  7EFE  2028  .(-......4.#...(
10C950   030C  104F  DD7E  05B9  3801  79DD  7705  1106  ........8.......
10C960   00DD  193A  BD72  3C32  BD72  FE20  3008  1804  ...:..<2....0...
10C970   23FD  3400  10C8  CDFB  7621  803C  2220  4021  #.4......!.<".@!
10C980   FE75  CD27  5211  003D  CD57  52FE  01CA  2152  ...'R..=.WR...!R
10C990   FE01  8258  7C02  CA4C  7CED  5332  7622  3476  ...X...L..S2."4.
10C9A0   1140  3DCD  5A52  FE01  CA21  5279  3236  7611  .@=.ZR...!R.26..
10C9B0   803D  2137  76CD  5D52  FE01  CA21  5278  3258  .=!7...R...!R.2X
10C9C0   763A  E837  FE80  381D  2100  3E22  2040  2148  .:.7..8.!.>".@!H
10C9D0   75CD  2752  CD45  52CD  CE76  FE01  CA21  52FE  ..'R.ER......!R.
10C9E0   0D28  DE18  EF
                       21  003E  2220  4021  2475  CD27  .(...!.>".@!!..'
10C9F0   52AF  32AC  7332  9175  21FF  FF22  EE5D  CD70  R.2..2..!.."....
```

Fig. 1. The Sector Display Minus the Spaces

✓See List of Advertisers on page 386

"... if no printer is attached, or the driver program was omitted or overwritten, your machine will hang up."

tion if the C register contains a 1, which is the computer's way of saying that the comparison matched).

Otherwise, the computer continues on its merry way down the instructions and we wind up with a Printer Not Ready message. We could have told it to jump there if the C register did not contain a 1, (JR NC,7CA8H is a jump if not); this can be done by typing the same Mod instruction, pressing the right arrow key 11 times and then pressing zero once, changing the line from: 381D, or in assembled form: 381D JR C,7CA8H, to: 301D, or in assembled form: 7C89 301D JR NC,7CA8H.

We changed the 38, which the computer understands as "jump if", to a 30, which tells it to "jump if not"—1D(hex) or 29 address locations, depending, in this case, on the value in C. If we had done it that way, and later hooked up a parallel printer, the value would have to be changed back again or we would get a Not Ready message anyway.

If you want to do serial printing under the display/edit mode, (H)ardcopy option of Profile, one more change has to be made. The code contained in the Access program looks like this:

```
7311 3AE837   LD     A,(37E8H)
7314 FE80     CP     80H
7316 3809     JR     C,7321H
7318 21D372   LD     HL,72D3H
731B CD8F72   CALL   728FH
731E C36471   JP     7164H
```

Still using NEWDOS and Superzap's DD option, select the drive number that the backup copy of Profile is in. For my copy, the track number was 1C, the sector was 3. If it isn't on yours, just match up the sector display with Fig. 2.

Locate the hex digits beginning with 3A (these are the numbers in the second through eighth columns of the listing, the group of six numbers in the first column is the hex address of the program location on the disk) and ending with 71. Fill these 32 places with zeros, and press Enter and Y to make the modification permanent.

You should now have a copy of Profile that will run with a serial printer. However, you should note that if no printer is attached, or the driver program was omitted or overwritten, your machine will hang up.

That wasn't so difficult, was it? And I really don't know how to program in machine language. All it took was a burning desire to get printed output, and some plowing into the Editor/Assembler manual. (And perhaps a truckload of perseverance!)

Now for T-BUG Users

It should be a lot easier for you, but I haven't tried to do it this way. Load the print program. Hold down the Break key and press the reset button. Get into the System mode and load and run T-BUG.

PRINT loads from 7000H to 7CFEH, T-BUG uses 4000H to 42E8H, so there is no conflict of memory. When the asterisk appears, type M 7C84. You should see: M 7C84 3A. Now type 00. The next memory location should be displayed: 7C85 E8. Again type 00 and keep doing that until you've gotten to memory location 7CA7. This is the last memory location you will zero out.

Press Break, put a blank tape in the cassette drive, set it up for recording and press P, T-BUG's Punch command, used to write memory locations to cassette. We need now specify a start address, end address and entry point, plus a name. The start address is 7000, the ending address is 7CFE. Here's where the fun starts.

According to NEWDOS, the entry point is 402D. Normally we would assume that a program's starting point and entry point are the same, but not necessarily. You'll have to try it both ways. Use Print as the program name. The punch line will look like this: P 7000 7CFE 402D PRINT, or P 7000 7CFE 7000 PRINT, depending on which entry ad-

```
11C300    C370  71ED  4BEE  5DCD  4B52  C3D2  71
                                             3A   E837  ....K...KR...:.7
11C310    FE80  3809  21D3  72CD  8F72  C364  71
                                             3A   2940  ..8.!........:)@
11C320    FE32  3805  3E0C  CD3B  0006  0B21  403C  0140  .28.>..;...!@<.@
11C330    3273  C511  7073  0140  00ED  B0E5  EB2B  0640  2......@.....+.@
11C340    7EFE  2020  082B  10F8  2170  7306  0123  0436  .....+..!....#.6
11C350    0D21  7073  7ECD  3B00  2310  F9E1  C110  D306  .!....;.#.......
11C360    053E  0ACD  3B00  3E0D  CD3B  0010  F4C3  7071  .>..;.>..;......
11C370    0182  B173  3AF1  5DB7  2009  21C6  72CD  8F72  ....:.....!.....
11C380    C364  7172  C03F  2220  4021  1C5E  CD27  52CD  ...!.?".@!...'R.
11C390    4552  FE59  2804  FE4E  20F5  CD00  67FE  4ECA  ER.Y(..N......N.
11C9A0    7071  3AEF  5A3D  4721  9D55  36C0  2310  FBED  ..:.Z=G!.U6.#...
11C9B0    4BEE  5DCD  4E52  2AED  5A23  22ED  5A21  9D56  K...NR*.Z#".Z!.V
11C9C0    CD2A  52CD  2752  AF32  F15D  3E01  32E9  5DC3  .*R.'R.2..>.2...
11C9D0    6471  1140  3FCD  5752  FE01  CA64  71FE  02CA  ...@?.WR........
11C9E0    6471  ED53  0D71  220B  7111  803F  CD5A  52FE  ...S..".....?.ZR.
11C9F0    01CA  6471  0132  3174  7932  1071  11C0  3F21  .....21..2....?!
```

Fig. 2. The Sector Display for ACCESS Minus the Spaces

THE PATCH

WE'VE MOVED!

Now we can offer faster and more reliable service -- Including UPS upon request!

DUALCASE*

UPPER/lowercase, full time from power-up; **NO** software; Standard typewriter keyboard operation (shift to UPPERCASE); Control characters can be displayed; 128 Total character set plus full graphics.

SWITCHABLE*

No switches. No driver. Operates from keyboard. Totally disables "PATCH" functions.

KEYBOARD DEBOUNCE*

Extra keyboard debounce, full time from power-up; **NO** software. If dirty keys are a problem, this is for you.

BLOCK CURSOR*

Replaces the underline style cursor directly. Easier to locate on a full screen. No distracting blinking.

SHORT CASSETTE LEADER*

For tape based systems. Does **NOT** change baud rate. Only shortens recorded leader. Saves four seconds of waiting time. Great for data files! No conflict with high baud rate cassette systems.

ELECTRONIC SHIFT-LOCK*

No extra keys or switches. Simply tap either shift key, UPPERCASE lock, normal shift unlocks.

"THE PATCH" is not only a lowercase modification, but a complete system offering many of the enhancements you would like available on your TRS-80. How is this possible? "THE PATCH" is a "smart chip" which monitors the same address lines as the ROM in your TRS-80 Model I Level II. "THE PATCH" recognizes certain addresses, turns off the LII ROM, then substitutes its' own byte of data. This "Patching" is what allows ALL these enhancements with **NO** software overhead, thus **NO** software conflict. In fact "THE PATCH" is the ONLY lowercase modification that is compatible with both Basic and machine language programs AND the ONLY all hardware lowercase modification.

You install "THE PATCH", no need to send your TRS-80 away so there's no hidden installation cost. Detailed instructions guide even the most inexperienced owner to complete installation in about 30 minutes. Works with the two or three chip ROM set. "THE PATCH" unit plugs into the ROM sockets (does not replace existing ROM). Lowercase does require installation of the extra video RAM (supplied with "THE PATCH").

"THE PATCH" is compatible with any word processor, any DOS and also other languages which use ROM sub-routines. "THE PATCH" is never obsolete. As new features become available they can be installed in your "PATCH".

"THE PATCH" is covered with a one year limited warranty on materials and workmanship. (Does void Radio Shack's 90 day warranty). Ask for details . . . we can install for a small fee.

✓62

CALL NOW
(208) 772-9571

CECDAT, INC.
P. O. Box 497
Hayden Lake, ID 83835

TRS-80 is a Trademark of Tandy Corp.

ORDER INFORMATION

"THE PATCH" is a trademark of CECDAT, INC.

Name ....

Street ....

City .... State .... ZIP ....

Signature ....

☐ Check, Money Order, Bank Draft

☐ VISA, MASTERCHARGE, Purchase Orders (add 3%)

Card/PO No. ....

Expiration Date .... Today's Date ....

You must check one:

☐ "MEM SIZE" ☐ "MEMORY SIZE"

Price valid through December 31, 1981

"THE PATCH 2" Switchable @ $94.97 ea. ....

OPTIONS: (please check)

☐ "Electronic Shift-Lock"	Add $30.00	....
☐ "Short Cassette Leader"	Add $10.00	....
☐ "Block Cursor"	No Charge	
☐ "Debounce"	No Charge	
Ship. & Hand. @ 2.50 ea.		....
COD ADD 5%		....
ID Sales Tax 3% (Id Res)		....
TOTAL ORDER		....

✓See List of Advertisers on page 386

"Always experiment on backup copies of data... Tattoo that on your arm somewhere."

dress you've used. Press Enter and the tape should start recording the program. Now go back to DOS and run TAPEDISK to create a disk file with the same Print.

When you've done that, or if you find that the entry address doesn't work, turn the expansion interface and the CPU keyboard off. Take any disks out of the drives first; this will ensure that all memory locations start up clear. You should then repeat the procedure for Access, modifying addresses 7311 to 7320 with 00 as in Print. Again we are faced with the entry point problem, giving us the following punch lines: P 7000 7460 402D ACCESS or P 7000 7460 7000 ACCESS. You don't have to press Enter this time, since the file name is six characters long. Make sure there is a blank tape in the cassette recorder set for record before you type this out. T-BUG starts the process automatically. Run TAPEDISK again, creating the file Access.

When you create the disk file, *do not* use an extension (/CMD, /BAS, whatever) on the file name. You may specify the destination drive. When you're creating the tape file, make sure you start the tape on the tape and not on the leader. I only mention this because I've done it myself. There you have it. See, it was easier than using NEWDOS (maybe). Remember, even hate mail costs money to send.

Data Manipulation

You still can't specify a range of data within the data base. (Please remember that although Microfiles is advertised as creating a data base, it actually compiles only a data set, not something that can be accessed, and the data manipulated, used, or modified by an alien program source as may a true data base.) Nor can the data be acted upon in any way but the ways imposed by Profile. But hang in there.

When Profile initializes the disk it grabs up all the available space it can and formats the disk as if it had files to store there. Take a look at Fig. 3. If you get the chance to look at a formatted diskette, you will notice that the only data is coded E5. See the difference? Think of the asterisks as claim markers that Profile uses to mark off disk space. It's a shame all those neat rows of asterisks have to be spoiled by the periods.

Notice that the data stored is 2A, and as long as there is a 2A there is an asterisk. When we have a period, there is an OD. If we realize that 2A is a hexadecimal number whose decimal equivalent is 42 ($2 \cdot 16 + 1 \cdot 10$), and then look at page C/2 in the Level II manual, we would see that 42 is the ASCII decimal code for an asterisk. OD, on the other hand, is decimal 13 ($16 \cdot 0 + 1 \cdot 13$), and that's a carriage return (where there is an ASCII character available for the code displayed, we see it; where there is none, Superzap places a period in the spot). Wouldn't it be nice if there was a function in Basic that creates a string like that, and something in Disk Basic that handles such a file? Oh, but there is! Ever hear of line input? Let me quote to you those blessed words from the Book of DOS, chapter 7, verse 42:

LINEINPUT (or LINE INPUT—the space is optional) reads everything from the first character up to:

1) an <EN> character not preceded by <LF>
2) the end-of-file
3) the 255th data character (this 255th character is included in the string)

Although the TRS-80 generates a line feed (LF) with a carriage return, only the carriage return is stored (OA is the hex value of a line feed). We seem to satisfy the requirements of line input for data retrieval from disk and at the same time, answer the burning question why we cannot store more than 255 characters.

Take a look at Fig. 4. It shows a sample set of data filed using the fields we specified at the beginning of the article. As usual, the first column of six numbers is the disk storage location of the data. If we go back for a moment to the file we set up and count the total elements in all the fields, we'll find our file length is 63 characters. If we then count out 63 hex numbers (each hexadecimal number is a pair, so the first line contains 16 numbers, the first of which is 49, followed by 4E, 20 and so on), at the end of 63 of them we find our friend OD. To read this data all we have to do is write a program:

```
100    OPEN "I", 1, "PRODAT:O":REM IF PRODAT IS ON
       DRIVE O
110    PRINT"DO YOU WISH TO ACCESS A FILE?"
120    A$ = INKEY$:IF A$<>"Y" AND A$<>"N" THEN
       120
130    IF A$ = "N" THEN 210
140    IF EOF(1) THEN PRINT "END OF FILE
       REACHED":GOTO 220
150    LINE INPUT#1, DATA$
160    IF LEFT$(DATA$,1) = "*" THEN PRINT "END OF
       DATA":GOTO210
170    IF LEFT$(DATA$,1) = CHR$(192) THEN PRINT
       "DELETED FILE": FOR X = 1 TO 500:NEXT X:CLS:
       GOTO 140
180    PRINT DATA$
190    PRINT"ANOTHER ?"
200    A$ = INKEY$:IF A$<>"Y" AND A$<>"N" THEN
       200
210    IF A$ = "Y" THEN CLS:GOTO 140
220    CLS:CLOSE:END
```

This is the skeleton of a sequential access program that can be used to read the disk data. We can customize the program to search for a particular value contained in the 64 character string (63 data characters plus the carriage return) using MID$. To handle the data, we would again use the MID$ function to break down the main string into substrings. We could then convert any numeric data we need from string form with the VAL function. Unfortunately, it's not simple to use sequential access to write to the disk file. We would have to input all the data, modify it, and then rewrite the entire file back to disk again. Otherwise, writing to disk under that mode would set the pointer back to the beginning of the file and all stored information would be lost. Always experiment on backup copies of data. If you make a mistake, the data will still be available. Tattoo that on your arm somewhere.

Suppose we wish to write some modified data to the file without entering Profile, using random access techniques. When we deal with random access, we retrieve or store data with the GET and PUT commands 256 character buffers. When we deal with less data than would fill a buffer, we can form the small data packets into a large group, using an array that will fill the buffer more efficiently. Using the same array, we can also retrieve the data.

```
106100    2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A2A ****************
106110    2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A2A ****************
106120    2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A2A ****************
106130    2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A2A  2A0D ***************
```

Fig. 3. The INFOFILE acquires disk space by filling vacant files with asterisks.

```
116500    494E  2048  4F43  4B20  434F  2E20  494E  432E IN.HOCK.CO..INC.
116510    2020  2020  2020  2030  3535  3231  3130  362F .......05521106/
116520    3132  2F38  3030  3031  382E  3437  2020  4E2F 12/800063.68..N/
116530    4120  202E  2E2E  2E2E  2E2E  2E2E  2E2E  2E0D A...............
```

Fig. 4. 63-Character File

"None of the MK functions allow for the storage of a single byte; it must be converted using ASCII."

By accident, our file contains a total of 63 characters plus the CR. Coincidentally, 64 can be divided into 256 an even four times. If we planned to use random techniques, we should make sure our data adds up to a number that, plus 1, will be evenly divisible into 256, as the Profile manual suggests. That means 3, 7, 15, 31, 63, 127 or 255. (Profile files are dumped to disk one after the other. If we try to access a file that does not conform to the above lengths we will find each buffer's remainder contains data which will be left out of the next GET call.) We could use a program like this:

```
100 Open "R", 1, "PRODAT:0"
120 EF = LOF(1)
130 FOR PF = 0 TO 3
140 FIELD 1, (PF*64) AS DUMMFILE$, 63 AS DATA$
150 NEXT PF
160 INPUT"FILE NUMBER: ";N
170 IF N> EF*4 THEN PRINT "OUT OF RANGE":FOR
    X = 1 TO 500: NEXT X:CLS: GOTO 160
180 FP = ((N-1)/4) + 1
190 PN = N-((FP*4)-3)
200 GET 1, FP
210 IF LEFT$(DATA$(PN),1) = CHR$(192) THEN PRINT
    "DELETED FILE":FOR X = 1 TO 500: NEXT X:CLS:
    GOTO 160:REM CHR$(192) IS USED TO FILL THE
    POSITIONS IN A DELETED FILE
220 IF LEFT$(DATA$(PN),1) = "#" THEN PRINT "NO
    SUCH FILE": FOR X = 1 TO 500: NEXT X:CLS:
    GOTO 160
230 PRINT DATA$(PN)
240 PRINT"ANOTHER...?"
250 A$ = INKEY$:IF A$<>"Y" AND A$<>"N" THEN 250
260 IF A$ = "Y" THEN CLS:GOTO 160
270 CLS: CLOSE: END
```

Again, it's a skeleton program. Flesh it out for specific fields, to change drives if more than one is used for storage and to search for particular pieces of data in string.

Infofile

Did I just say "change drives"? How the heck can the program know if I have more than one drive? Allow me to introduce you to Fig. 5a. This is a section of a diskograph for another file created by Profile, called Infofile.

The first pair of hex digits (a hex pair is called a byte) 18, convert to decimal 24. It just so happens that in this file I have stored 24 records. In fact, the first two bytes in this record of Infofile contain the number of records currently in the Prodat file. It is saved to disk using the MKI$ in a random file mode, since it will always be an integer. If it had a numeric value larger than one byte (not simply 1800, which has no value in the second byte), it would not necessarily have been directly readable without using the alternate CVI function to reconvert it to a numerical value.

The next two bytes (A007) are the MKI$ converted value for the maximum allowable number of files, followed by the single byte, 01, which is the maximum drive number. (This value is stored as CHR$(n) where n is the maximum drive number. None of the MK functions allow for the storage of a single byte; it must be converted using ASC.) The two bytes, 0000, note that there are no deleted records in this file. Had there been any, their number would have been stored here. The next piece of data, 40, is the length of the record, 64 (63 characters for the file data plus the carriage return). As with the maximum drive number and the following byte, 07, (which is the number of fields assigned at initialization) this was stored as a CHR$(n) and can be converted to its numeric value with the ASC function. The last four byte pairs tell us what the highest file number is on drives 0, 1, 2 and 3 respectively.

Fig. 5b is the second sector of Infofile, showing our field names and a small amount of additional data. The first 13 bytes are allotted to the field name itself, while bytes 14 and 15 are the MKI$ equivalent of the screen print position, plus 15360. (Therefore, if printing to the screen we would use Print @, CVI(x$)-15360.) Byte 16 is the length of the field. If we look at the

```
F00000     1800  A007  0100  0040  077B  029F  0700  0000 .........@.......
F00010     00DC  559D  0500  0000  0000  0000  0000  0000 ..U.............
```

Fig. 5a. The first set of numbers, 18, is the hex equivalent of 24, the total number of files on record.

```
F00100     4143  434F  554E  5420  4E41  4D45  204E  3C17 ACCOUNT.NAME.N<.
F00110     494E  564F  4943  4523  2020  2020  208E  3C06 INVOICE#......<.
F00120     4441  5445  2020  2020  2020  2020  20CE  3C08 DATE..........<.
F00130     414D  4F55  4E54  2020  2020  2020  200E  3D07 AMOUNT........=.
F00140     504F  2F4A  4F42  2320  2020  2020  204E  3D07 PO/JOB#......N=.
F00150     504F  5354  4544  2F50  4149  4420  208E  3D08 POSTED/PAID...=.
F00160     4348  4543  4B23  2020  2020  2020  20CE  3D04 CHECK#........=.
```

Fig. 5b. The Second Sector of INFOFILE, Showing the Titles of the Established Fields

"The user's manual gives a great program for extracting this data from a specified file number."

line that contains the field Account Name, we see the final byte is 17, 23 decimal, the length of our entry for that field.

Procedures

The first thing we must do is define some variables. We'll use the ones that are provided in the Profile user's manual:

NR	The Number of Records in Prodat, including any deleted records
MX	The Maximum Number of Records Allowed
MD	The Maximum Drive Number
NF	The Number of Fields in the Record
D(I)	"I" is an integer between 0 and 3. D(I) is the highest record number stored on disk drive "I"

And now some program lines:

```
100 CLS: CLEAR 1000
110 OPEN "R",1, "INFOFILE:0"
120 FIELD 1, 2 AS NR$, 2 AS MX$, 1 AS MD$, 2 AS
    DR$, 1 AS RL$, 1 AS NF$, 2 AS D$(0), 2 AS D$(1),
    2 AS D$(1)
    2 AS D$(2), 2 AS D$(3)
```

Usually when dealing with a random access file, we'd use an array to break down the elements in the 256 character buffer. However, here we are only interested in the elements we need for the program.

```
130 GET 1,1
140 NR = CVI(NR$): MX = CVI(MX$): MD = ASC(MD$):
    NF = ASC(NF$): FOR DN = 0 TO 3: D(DN) = CVI(D$
    (DN)): NEXT DN
```

Normally, we would then get the second buffer and discover the field names and lengths. The user's manual gives a great program for extracting this data from a specified file number. As we already know these things, we can bypass that section.

```
150 CLOSE 1
```

We will assume that the first 90 lines of the program load all the account names and addresses into a matrix in memory, MASTERLIST$, which will be used to coordinate the data from PRODAT. (Otherwise, we should have three disks on line, two for Profile and one for the account data.) In the program lines that follow, we will use the variable EF to indicate the numeric position of the final entry in MASTERLIST$ and the variable RP to indicate any intermediate relative position within the matrix.

```
160 DN = 0 :DN$ = STR$(DN)
170 OPEN "R",1, "PRODAT:";DN$
180 FOR I = 0 TO 3
190 FIELD 1, (I*63) AS DUMMYSTRING$, 23 AS NM$(I),
    6 AS IN$(I), 8 AS DT$(I), 7 AS AM$(I), 7 AS PO$(I)
    8 AS PD$(I), 4 AS CK$(I)
200 NEXT I
```

Now use the data in Masterfile$ to check out the status of the accounts stored in PRODAT.

```
210 FOR RP = 1 TO EF
220 FOR BN = 1 TO INT(D(DN)/4)
230 GET 1, BN
240 FOR I = 0 TO 3
250 IF LEFT$(NM$(I),1) = CHR$(42) THEN 280
260 IF LEFT$(NM$(I),1) = CHR$(192) THEN 1000
270 IF NM$(I) = MASTERFILE$(RP,1) AND LEFT$(PD$(I),1)
    = "." THEN 2000
280 NEXT I, BN
```

Line 250 procedes to the next record in the subfile if the entry being currently examined was deleted. Line 260 sends the program to a controlled error message if the end of the stored data has been reached and no information has been recovered for a given account name.

```
290 DN = DN = 1: DN$ = STR$(DN): IF DN>MD THEN 310
300 CLOSE 1: OPEN "R", 1, "PRODAT:";DN$:GOTO 220
310 IF DN<>0 THEN DN = 0: CLOSE 1: DN$ = STR$(DN):
    OPEN "R", 1, "PRODAT:";DN$
320 NEXT RP
```

The loose end, lines 2000-xxxx, is merely a print statement that will send the information to our line printer (formatted to reflect page number of account statement on a per account basis) and performs the addition necessary to give total due amounts.

Fig. 6 is a small program for single disk users that will read each of the Prodat entries over a given range of dates (for instance, if you wanted the entries for August you would use 08/00/80 in response to the Greater Than question and 09/00/80 for the Less Than prompt), print the account name, invoice number and amount with pagination and, when done, print the total amount involved. More fields for manipulation can be specified and other actions performed

Program Listing

```
10 CLEAR 1000
30 CLS:OPEN "R",1,"PRODAT:0"
50 EF=LOF(1)
60 DIM F$(EF)
70 FOR J=0 TO 3
90 FIELD 1, (64*J) AS DUMMY$, 63 AS F$(J)
110 NEXT J
115 PRINT @ 320,"MONTH OF : ";:INPUT M$:CLS
```

Program continues

Program continued

```
120 PRINT @ 320, "DATE RANGE:";:INPUT"GREATER THAN:
";GT$:PRINT @ 448,"";:INPUT"LESS THAN: ";LT$
130 CLS
180 FP=1
190 GET 1,FP
195 FOR RP=0TO3
197 IF LEFT$(F$(RP),1)=CHR$(42) THEN 290
199 IF LEFT$(F$(RP),1)=CHR$(192) THEN PRINT T:GOTO 202
200 IF RIGHT$(F$(RP),4)="...." AND
MID$(F$(RP),30,8)>GT$ AND MID$(F$(RP),30,8)<LT$ THEN
GOSUB 400 ELSE 202
201 N$=LEFT$(F$(RP),23):IN$=MID$(F$(RP),24,6):AM=
VAL(MID$(F$(RP),38,7)) : T=T+AM:LPRINTN$;"
";IN$;"          ";:LPRINT USING"####.##";AM: LS=LS+1:
IF LS>50 THEN 500
202 NEXT RP
204 FP=FP+1:IF FP>EF THEN 290
210 GOTO 190
290 LPRINT "
  ";"======="
300 LPRINT "
";:LPRINT USING"$$####.##";T
310 CLOSE
320 END
400 IF FL>0 THEN RETURN
410 PN=PN+1:FOR X=1TO2: LPRINTCHR$(13); :NEXTX
:LPRINT"COPYQUICK<LF>107 EAST 42ND<LF><LF>
   CHARGE DATA    MONTH OF ";M$;"
PAGE ";
415 LPRINT USING"###";PN:LPRINT"<LF>";:LPRINT"ACCOUNT
NAME                    INVOICE#
AMOUNT":FL=1:RETURN
500 FORX=1TO6:LPRINTCHR$(13);:NEXTX:FL=0:GOTO202
```

on the data by simply following the pattern the program provides.

That's it. Those few lines are all it takes to turn Profile into more than just an overpriced in-memory information system for disks. We now have available to us a feature seen on many, if not all, of the large multiuser computers: an accessible data base of information! For a relatively small machine such as the TRS-80 that is some accomplishment!

If you're not a business and have no accounts receivable to take care of, don't think you can escape Profile. Do you collect comics? Magazines? Computer Programs? What about all you CB and amateur radio operators? Want your logs up to date?

Forget about Microfiles. Its' expanded entry formatting is very nice if all you want to do is enter data and recall it according to the parameters present in the program itself. But what good is collecting information if you can't do whatever you want with it?

Profile gives you this added flexibility, and at a cheaper price. (Isn't that against the law?)■

✓See List of Advertisers on page 386

A home inventory program that remembers the little things.

Wordly Goods

John E. Fail
6170 Downey Ave.
Long Beach, CA 90805

Have you ever wondered about the possibility of your house being burglarized or burning down? We live with this possibility every day whether we are at home or not. In the event of such a catastrophe your insurance should cover the loss. However, when the insurance man comes to call, will you be able to accurately recall all the items you have lost? Chances are you will remember the tv set, the couch, the computer, the major items in your household; it is the little items you will not easily recall—tools, pictures, jewelry and the like.

This program, for a 48K system with two disk drives, catalogs 300 items per room and provides a line printer routine.

Depending on the total items in your inventory, you could utilize one file for all items in your household. Included in the program are full correction and deletion routines. Categories included in an individual file are: location; item; data purchased; initial cost; business the item was purchased from; serial number; and comments.

```
Fail Family Household Inventory

Directory
1 = Load Data From Disk
2 = Look at List for Living Room
3 = Save Data on Disk
4 = Printout of Data on File
5 = Delete an Entire Entry
6 = Correct an Entry
7 = Add Entries to File
8 = Print Total Value of Living Room
9 = Item Search by Name
Your Selection Please?
```

Fig. 1 Directory Layout

The program is straightforward, user oriented and prompting at all points. The scheme is to set up individual files on a separate disk by room. The other alternative is to file your inventory on the program disk providing the inventory is not too large. The inventory program can be used by 32K systems by adjusting the clear statement in line 20 and the dimension statements in line 25. Line 30 should be changed to fit your family name.

After you have run the program, entered files for your inventory and are ready to save to disk, you will be prompted by the program to provide a filespec for writing to disk. This list of filespecs will be provided just prior to loading from or to disk. The user should change lines 13010 and 13020 to his or her own filespec if those in the program are not acceptable.

A word of caution—plenty of string space is provided in the clear statement in line 20, however, don't get too carried away with large comments or string space will be used up fast.

When first running the program and entering files, the first item entered will call for a location. No further locations will be asked for during the course of the entries. The program will automatically print that location in all entries after number one. If you are not using separate files for all rooms in the house but use one file for the entire household, a single filespec will be used to answer the location question. On the first entry use a word such as "household." If

```
                  Inventory for Living Room
Number  Item                                      Cost      Mfr/Dealer   Date
  1.    Couch                                     $ 995.00  Sears        02/05/81
        Serial Number: None
        Comments: Gold with Flower Patterns
  2.    Chair                                     $ 140.00  Sears        03/01/81
        Serial Number: None
        Comments: Gold with Flower Pattern (Set of 4)
  3.    Television                                $1000.00  Dooleys      04/01/80
        Serial Number: 43566981
        Comments: Motorola Model 333-8799.1
  4.    Lamp Floor                                $  40.00  Sears        01/01/81
        Serial Number: None
        Comments: With Shade
             ••••Continue 'C' Finished 'F' ••••? _
```

Fig. 2 Print List Sample

you are using separate files for each room, assign that room's name to the location question and separate the rooms by filespec when saving and loading from disk.

Search, Delete and Correct

The search section is based on key words. When searching for an item, the item variable is compared against the first three letters of the search request input. For example, suppose you have three chairs in a room. When you are requested for an item to search for, enter chair. The program will then search out the first chair encountered and print it out. The program will then ask if this is the correct chair; if not, the program will continue the search. If no comparison is found, the program will so advise you. When entering data into the item section, use the major subject first, such as chair, couch, television, computer. If you still cannot find the item you are looking for there is always the 'Look At List' function on the menu to locate an item.

The deletion routine is very straightforward. When an item is deleted, all other items are moved up one to fill the space left by the deleted item. When the deletion routine is called, the program will check with you to be certain you are indeed deleting the item you wish to delete.

The correction routine operates much like the deletion routine. It checks to make sure this is the item you wish to correct, and then displays an input form similar to the normal entry input routine. You need only enter the portion to be corrected. If the item, cost and dealer information is correct and you only want to enter a new serial number, merely press Enter for the items you do not want to change and enter the new data in the serial variable. All information in the other columns will remain unchanged.

The Line Printer Routine

The line printer routine is set up for 80-column printers to maintain compatibility with most any printer in use these days. The printer function is very handy for keeping a permanent record of the inventory. You can run out a copy of each listing and keep it in a safe place, such as a friend's home, your place of employment, or a safe deposit box. This ensures the list will not disappear with the house or burglar. It may be a good idea to keep a separate copy of the program disk and data disk outside the home also. The listing automatically prints "a value of room routine" at the end of the printout. This function is also available on the video for a quick check of the total value without a printout.

Readers without line printers should remove the 4000 series of line numbers to prevent locking up the program in case of inadvertently going to that function. This program can also be used by cassette-based systems merely by changing the 1000 and 3000 series of line numbers to accomodate cassette I/O.■

Inventory for Living Room

Page: 1 Last Revision: 6 April 1981

Number	Item	Cost	Mfr/Dealer	Date	Serial Number
1.	Couch	$ 995.00	Sears	02/05/81	None
	Comments: Gold with Flower Patterns				
2.	Chair	$ 140.00	Sears	03/01/81	None
	Comments: Gold with Flower Pattern (set of 4)				
3.	Television	$1000.00	Dooleys	04/01/80	43566981
	Comments: Motorola Model 333-8799.1				
4.	Lamp Floor	$ 40.00	Sears	01/01/81	None
	Comments: With Shade				
5.	Lamp	$ 100.00	Wards	02/02/81	None
	Comments: Matched set of 2 Table Lamps				
6.	Clock Digital	$ 30.00	Sears	12/25/80	322479
	Comments: With Alarm and FM Radio				
7.	Table	$ 140.00	Sears	02/02/81	None
	Comments: Coffee Table with Glass Top				
8.	Stereo System	$ 600.00	Wards	09/18/81	5559091
	Comments: With 50-60 Records in Cabinet				
9.	Painting	$9999.00	Art Shop	01/01/81	None
	Comments: The Mona Lisa Original by Da Vinci				

Total Value of Living Room $13,044.00

Fig. 3 Line Printer Print Sample

Program Listing

```
20 CLEAR25000:T=300:U$="$####.##":V$="$##,###.##"
25 DIML$(300),I$(300),C(300),D$(300),M$(300),S$(300),CO$(300)
30 B$=L$(1):SK=0:SW=0:CLS:PRINTTAB(15)"FAIL FAMILY HOUSEHOLD INV
ENTORY":PRINTSTRING$(63,"="):PRINTTAB(24)"DIRECTORY":PRINTSTRING
$(63,"="):PRINT"1 = LOAD DATA FROM DISK":PRINT"2 = LOOK AT LIST"

35 PRINT@337,"FOR ";B$
40 PRINT"3 = SAVE DATA ON DISK":PRINT"4 = PRINTOUT OF DATA ON FI
LE":PRINT"5 = DELETE AN ENTIRE ENTRY":PRINT"6 = CORRECT AN ENTRY
"
50 PRINT"7 = ADD ENTRIES TO FILE":PRINT"8 = PRINT TOTAL VALUE OF
"
55 PRINT@729,B$
57 PRINT@768,"9 = ITEM SEARCH BY NAME"
60 PRINT@832,STRING$(63,"=")
70 PRINT:PRINT@896,"YOUR SELECTION PLEASE";:INPUTA:IFA<1ORA>9GOT
O30
80 ONAGOTO1000,2000,3000,4000,5000,6000,7000,8000,9000:GOTO30
1000 'LOAD FROM DISK ROUTINE
1020 GOSUB13000
1025 CLS:PRINTTAB(20)"LOAD FROM DISK ROUTINE":PRINT@448,"PLACE D
ATA DISK IN DRIVE AND PRESS ENTER";:INPUTA$
1029 ONERRORGOTO14020
1030 OPEN"I",1,FS$
1040 CLS:PRINT@448,"LOADING FILE #"
1100 FORL=1TOT
1110 PRINT@462,L
1160 INPUT#1,L$(L),I$(L),C(L),M$(L),D$(L),S$(L),CO$(L)
1180 IFEOF(1)THEN1980
1500 NEXTL
1980 CLOSE1
1990 B$=L$(1)
1999 GOTO30
2000 'LOOK AT LIST ROUTINE
2010 B$=L$(1)
2020 GOSUB12500
2025 H=0
2030 FORL=1TOT
2035 IFLEN(L$(L))=0THEN12700
```

Program continues

Program continued

```
2036 IFSK=1ANDLEN(L$(L))=0THENPRINT@448,"NO DATA ON FILE FOR ITE
M #";ID;" PRESS ENTER";:INPUTA$:GOTO5100
2038 IFSW=1ANDLEN(L$(L))=0THENPRINT@448,"NO DATA ON FILE FOR ITE
M # ";ID;" PRESS ENTER";:INPUTA$:GOTO6080
2040 PRINTL".";TAB(7)I$(L);:PRINTTAB(29)USINGU$;C(L);:PRINTTAB(4
1)M$(L);TAB(54)D$(L):PRINTTAB(7)"SERIAL NUMBER: "S$(L):PRINTTAB(
7)"COMMENTS: "CO$(L):H=H+3:IFH=12GOSUB12580
2045 IFSK=1GOTO5050
2050 IFSW=1GOTO6050
2800 NEXTL
2999 GOTO30
3000 'OUTPUT TO DISK ROUTINE
3005 GOSUB13000
3010 CLS:PRINTTAB(15)"OUTPUT DATA TO DISK ROUTINE":PRINT@448,"PL
ACE DATA DISK IN DRIVE AND PRESS ENTER";:INPUTA$
3020 CLS:PRINT@448,"SAVING FILE #"
3025 OPEN"O",1,FS$
3030 FORL=1TOT
3040 PRINT@461,L
3070 IFLEN(I$(L))=0THEN3998
3075 L$(L)=B$
3080 PRINT#1,CHR$(34);L$(L);CHR$(34);CHR$(34);I$(L);CHR$(34);C(L
);CHR$(34);M$(L);CHR$(34);CHR$(34);D$(L);CHR$(34)
3090 PRINT#1,CHR$(34);S$(L);CHR$(34);CHR$(34);CO$(L);CHR$(34)
3500 NEXTL
3998 CLOSE(1)
3999 GOTO30
4000 'LINE PRINT ROUTINE
4010 CLS:PRINTTAB(15)"LINE PRINTER PRINTOUT ROUTINE":PRINT:PRINT
"ENTER THE DATE TODAY";:INPUTDA$
4015 PRINT:PRINT"SET PRINTER TO WIDE CHARACTERS AND PRESS ENTER"
;:INPUTLP$
4020 CLS:PG=0:GOSUB12800
4025 PRINT@640,"PRINTING ITEM #:"
4030 FORL=1TOT
4035 PRINT@656,L
4040 IFLEN(L$(L))=0THEN4230
4060 LPRINTL;".";TAB(6)I$(L);:LPRINTTAB(28)USINGU$;C(L);:LPRINTT
AB(38)M$(L);TAB(53)D$(L);TAB(63)S$(L):LC=LC+1:GOSUB14000
4065 LPRINTTAB(6)"COMMENTS: "CO$(L):LC=LC+1:GOSUB14000
4070 LPRINTSTRING$(79,"-"):LC=LC+1:GOSUB14000
4220 NEXTL
4230 SW=1:GOTO8000
4980 SW=0:LPRINT"":LPRINT"TOTAL VALUE OF ";B$;" ";:LPRINTUSINGV$
;RT:LPRINTCHR$(12)
4999 GOTO30
5000 'DELETE ENTRY ROUTINE
5010 SK=0:CLS:PRINTTAB(15)"DELETE ENTRIES ROUTINE":PRINT@448,"WH
AT IS THE ITEM NUMBER TO BE DELETED";:INPUTID
5015 IFID<1ORID>300THEN5010
5020 CLS:SK=1:L=ID:H=0:GOSUB12500
5040 GOTO2036
5050 PRINT@448,"CORRECT ITEM TO BE DELETED (YES/NO)";:INPUTA$
5060 IFLEFT$(A$,1)="N"THENSK=0:GOTO5100
5065 CLS:PRINT@448,"DELETEING ITEM #: ";ID
5070 FORX=IDTOT
5080 L$(X)=L$(X+1):I$(X)=I$(X+1):C(X)=C(X+1):M$(X)=M$(X+1):D$(X)
=D$(X+1):S$(X)=S$(X+1):CO$(X)=CO$(X+1)
5085 IFLEN(L$(X))=0THEN5100
5090 NEXTX
5100 CLS:PRINTTAB(15)"DELETE ENTRIES ROUTINE":PRINT@448,"DELETE
ANOTHER ITEM (YES/NO)";:INPUTDK$
5110 IFLEFT$(DK$,1)="Y"GOTO5010ELSESK=0:GOTO30
5999 GOTO30
6000 'CORRECT ENTRY ROUTINE
6010 CLS:PRINTTAB(15)"CORRECT ENTRY ROUTINE"
6020 PRINT@448,"WHAT IS ITEM NUMBER TO BE CORRECTED";:INPUTID
6025 IFID<1ORID>300THEN6010
6030 SW=1:GOSUB12500
6040 L=ID:H=0:GOTO2038
6050 PRINT@448,"CORRECT ITEM TO BE CORRECTED (YES/NO)";:INPUTA$
6060 IFLEFT$(A$,1)="Y"THENCLS:SW=1:L=ID:PRINT@192,"ITEM # ";ID:G
OTO7110
6070 IFLEFT$(A$,1)="N"THENSW=0:GOTO6080
6080 SW=1:CLS:PRINTTAB(15)"CORRECT ENTRY ROUTINE":PRINT@448,"CHA
NGE ANOTHER ITEM (YES/NO)";:INPUTDK$
6090 IFLEFT$(DK$,1)="Y"GOTO6010ELSESW=0:GOTO30
6999 GOTO30
7000 'ADD ENTRIES ROUTINE
7010 CLS
7050 FORL=1TOT
7100 IFLEN(L$(L))=0THENPRINTTAB(15)"ADD ENTRIES TO LIST ROUTINE"
:PRINTSTRING$(63,"="):PRINT@192,"ITEM #";LELSE7900
7105 PRINT@192,"ITEM #";L
7110 IFL=1THENPRINT@256,"ENTER ROOM:";:INPUTL$(L)
7115 IFL>1THENPRINT@256,L$(1)
7120 PRINT@320,"ITEM (19 MAX)";:INPUTI$(L)
```

Program continues

Color computer owners,

32K PLUS DISKS* $298.00

Yes, that's right - for as little as $298.00 you can add 32K of dynamic RAM, and a disk interface, to your TRS-80 Color Computer! If you just want the extra memory it's only $199.00, and you can add the disk interface later for $99.00.

Just plug the *Color Computer Interface (CCI)*, from Exatron, into your expansion socket and "Hey Presto!" - an extra 32K of memory. No modifications are needed to your computer, so you don't void your Radio Shack warranty, and Exatron give both a 30 day money-back guarantee and full 1 year repair warranty on their interface.

The *CCI* also contains a 2K machine-language monitor, with which you can examine (and change) memory, set break-points, set memory to a constant and block-move memory.

So what about the *CCI Disk Card*? Well as we said it's only an extra $99.00, but you'll probably want Exatron's *CCDOS* which is only $29.95 – unless you want to write your own operating system. The *CCI Disk Card* uses normal TRS-80 Model I type disk drives, and *CCDOS* will even load Model I TRSDOS disks into your color computer – so you can adapt existing TRS-80 BASIC programs.

As a further plus, with the optional *ROM Backup* adaptor, you can dump game cartridges to cassette or disk. Once the ROM cartridge is on cassette, or disk, you can reload, examine and modify the software. The *ROM Backup* adaptor is only $19.95.

For more information, or to place an order, phone Exatron on their Hot Line *800-538 8559* (inside California 408-737 7111), or clip the coupon.

excellence in electronics

exatron

DEALER ENQUIRIES INVITED

Exatron, 500
181 Commercial Street,
Sunnyvale, CA 94086

☐ Please send a 32K Color Computer Interface for $199.00

☐ Please send a CCI Disk Card for $99.00

☐ Please include CCDOS and manual for $29.95

☐ Also include a ROM Backup adaptor for $19.95

Please add $5.00 for shipping to all orders, and 6 percent sales tax in California.

Name ....

Address ....

City ....

State .... Zip ....

Charge my:

☐ MasterCard Interbank Code ....

☐ Visa Expiration Date ....

Card ....

☐ Check enclosed for ....

☐ Ship COD ($2.00 extra)

Signature ....

*32K RAM plus Disk Interface TRS-80 is a trademark of Tandy

Program continued

```
7130 IFLEN(I$(L))>19THENPRINT@335,STRING$(40," "):GOTO7120
7140 PRINT@384,"COST (7 MAX I.E. 9999.99)";:INPUTC(L)
7160 PRINT@448,"MANUFACTURER/DEALER (11 MAX)";:INPUTM$(L)
7180 IFLEN(M$(L))>11THENPRINT@478,STRING$(20," "):GOTO7160
7200 PRINT@512,"DATE PURCHASED (MM/DD/YY)";:INPUTD$(L)
7210 IFLEN(D$(L))>9THENPRINT@539,STRING$(20," "):GOTO7200
7220 PRINT@576,"SERIAL # (14 MAX)";:INPUTS$(L)
7230 IFLEN(S$(L))>14THENPRINT@594,STRING$(20," "):GOTO7220
7240 PRINT@640,"COMMENTS (40 MAX)";:INPUTCO$(L)
7250 IFLEN(CO$(L))>40THENPRINT@659,STRING$(45," "):GOTO7240
7300 PRINT@896,"IS ABOVE DATA CORRECT (YES/NO)";:INPUTF$:IFLEFT$
(F$,1)="Y"THENPRINT@896,STRING$(40," ")
7310 IFLEFT$(F$,1)="N"THENL=L:CLS:GOTO7105
7315 IFSW=1GOTO6080
7320 PRINT@896,"ENTER 'A' FOR ANOTHER ENTRY. 'R' RETURN TO DIREC
TORY";:INPUTG$
7340 IFG$="A"THEN7900ELSE30
7900 CLS:NEXTL
7999 GOTO30
8000 'TOTAL VALUE OF ROOM ROUTINE
8040 B$=L$(1)
8050 CLS:RT=0
8055 PRINT@448,"TOTALING VALUE OF ";B$
8070 FORL=1TOT
8090 RT=C(L)+RT
8130 NEXTL
8140 IFSW=1GOTO4980
8150 CLS:PRINT@448,"VALUE OF ";B$;" ";:PRINTUSINGV$;RT:PRINT@960
,"PRESS ENTER";:INPUTA$:GOTO30
8999 GOTO30
9000 ' SEARCH FOR ITEM ROUTINE
9010 GOSUB14050:PRINT@448," NAME OF THE ITEM YOU ARE SEARCHING F
OR";:INPUTSE$
9020 GOSUB14050:PRINT@448,"SEARCHING FOR ";SE$
9040 FORL=1TO301
9045 IFL=301ORLEN(I$(L))=0THEN9250
9050 IFLEFT$(I$(L),3)=LEFT$(SE$,3)THENGOSUB12500ELSE9200
9060 PRINTL".";TAB(7)I$(L);:PRINTTAB(29)USINGU$;C(L);:PRINTTAB(4
1)M$(L);TAB(54)D$(L):PRINTTAB(7)"SERIAL NUMBER: "S$(L):PRINTTAB(
7)"COMMENTS: "CO$(L)
9070 PRINT@448,"IS THIS THE ITEM YOU ARE LOOKING FOR (YES/NO)";:
INPUTSI$
9075 IFLEFT$(SI$,1)="Y"THEN9100
9080 IFLEFT$(SI$,1)="N"THENGOSUB14050:PRINT@448,"CONTINUE SEARCH
 FOR ";SE$" (YES/NO)";:INPUTSJ$
9090 IFLEFT$(SJ$,1)="N"THEN9100ELSE9200
9100 GOSUB14050:PRINT@448,"SEARCH FOR ANOTHER ITEM (YES/NO)";:IN
PUTSA$:IFLEFT$(SA$,1)="Y"THEN9010ELSE30
9200 NEXTL
9250 PRINT@448,STRING$(40," "):PRINT@448,"NO FURTHER DATA AND/OR
 DATA ON FILE FOR ";SE$:FORZ=1TO1000:NEXTZ:GOTO9100
9998 GOTO30
9999 END
12500 CLS:PRINTTAB(20)"INVENTORY FOR ";B$:PRINTTAB(1)"#";TAB(7)"
ITEM";TAB(29)"COST";TAB(41)"MFR/DEALER";TAB(55)"DATE":PRINTSTRIN
G$(63,"=")
12550 RETURN
12580 H=0:PRINT@980,"**** CONTINUE 'C' FINISHED 'F' ****";:INPUT
J$
12590 IFJ$="F"THEN30ELSEGOSUB12500
12600 RETURN
12700 PRINT@960,"END OF LIST FOR ";B$;" PRESS ENTER";:INPUTK$:GO
TO30
12800 LC=0:PG=PG+1:LPRINTTAB(25)"INVENTORY FOR ";B$:LC=LC+1:LPRI
NT" ":LC=LC+1:LPRINT"PAGE:"PG;TAB(45)"LAST REVISION: ";DA$:LC=LC
+1
12805 LPRINTSTRING$(79,"="):LC=LC+1:LPRINT" ":LC=LC+1
12810 LPRINTTAB(1)"#";TAB(6)"ITEM";TAB(28)"COST";TAB(38)"MFR/DEA
LER";TAB(54)"DATE";TAB(63)"SERIAL #":LC=LC+1:LPRINTSTRING$(79,"-
"):LC=LC+1
12820 RETURN
13000 CLS:PRINTTAB(20)"FILESPEC KEYWORD LIST"
13010 PRINT:PRINT"THE FOLLOWING FILESPEC WORDS ARE AVAILABLE:":P
RINT:PRINT"RADIO ROOM = RADIORM":PRINT"LIVING ROOM = LIVINGRM":P
RINT"DINING ROOM = DININGRM"
13020 PRINT"KITCHEN = KITCHEN":PRINT"MASTER BEDROOM = MASBEDRM":
PRINT"MIKES BEDROOM = MIKEROOM":PRINT"GARAGE = GARAGE":PRINT"PAT
IO = PATIO"
13030 PRINT@960,"FILESPEC DESIRED";:INPUTFS$
13040 RETURN
14000 IFLC=60THENLPRINTSTRING$(5,CHR$(10)):LC=0:GOSUB12800
14010 RETURN
14020 RESUME14030
14030 CLS:PRINT@460,"********** FILE SPEC ERROR **********":FORZ
=1TO1000:NEXTZ:GOTO30
14050 CLS:PRINTTAB(19)"SEARCH FOR ITEM ROUTINE":PRINTSTRING$(63,
"=")
14060 RETURN
```

Poor Man's Floppy

HIGH SPEED CASSETTE SYSTEM

FOR TRS-80*

Now the widely acclaimed JPC Cassette System is available for your TRS-80* computer. The price is only $90.00

[Reprint of June 1980 Review, *80 Microcomputing*]

TC-8 Cassette System
JPC Products
Albuquerque, NM
Kit: $90
Assembled: $120

by Carl A. Kollar

I guess I don't have to tell any TRS-80 owners how frustrating the cassette system that comes with the computer can be. Even with the factory mod that's available, the annoyance of loading and checking programs becomes just barely tolerable.

If you're like me, after you've just plunked down a chunk of money for a Level II 16K machine, "you ain't got nuttin left" for even one disk drive at 500 bucks apiece. So you suffer.

A reasonable alternative is the Exatron Stringy Floppy (ESF). This will cost you about 250 bucks and totally eliminates your loading and saving problems, automatically and fast. I've had one of these for about six months and love it!

But, if the price is still too steep, have I got a device for you!

The Device

The February 1980 issue of *Microcomputing* had an ad that intrigued the hell out of me. It was a high-speed cassette system by JPC Products acclaimed as a "poor man's floppy." It made all sorts of seemingly ridiculous claims such as "loads five times faster," "stores 50,000 bytes on a 10-minute cassette," "less than one bad load in a million bytes with the volume control anywhere between one and eight."

All this for a measly [90] bucks? How could this be? A call to Albuquerque answered a few questions: Yes, it had its own power supply, and, it stored programs five times faster because it utilized higher density data. The computer outputs the information at a higher rate out of the rear keyboard connector.

The ad had even claimed anyone could build it even if you have never soldered before. JPC would make it work, if you couldn't—for free. I was sold. I placed my order, and it arrived about two months later (parts shortage).

I work in electronics, so I found the unit exceptionally easy to build. It took about an hour. The manual is superb. (That's better than great.) It was clear, concise and exact with no ambiguities. Important parts placements are stressed (polarity markings on electrolytics, bands on diodes, etc.).

JPC was right! With these instructions, you couldn't go wrong. The board quality is excellent. It is double-sided and parts locations are clearly marked on the component side of the board. There are no jumper wires to install. JPC utilizes PC traces and plated-through holes for connections to traces on the other side of the board.

Also, there are absolutely no adjustments or settings to bother with.

The documentation is a sheaf of 8½ × 11 papers stapled together. It is written in the nicest format I've seen in a while. Each command and/or subjects is covered on its own sheet in large type. All explanations are in easy to read English—not computerese.

Commands and Features

SAVE"filename": Saves your BASIC program on cassette.
LOAD: Reads the next BASIC program from the cassette.
LOAD"filename": Searches for and loads the specified file from cassette.
LOAD? and LOAD?"filename": Reads file from cassette, and compares contents to memory.
LOADN: Prints a list of all the programs on a cassette, until interrupted by the "break" key.
LOADN"filename": Same as above except the tape will stop at the end of the program named.
KILL: Removes the file manager program from memory so that the extra memory can be used by large programs.
RSET: Allows the operator to rewind and position the tape on tape recorders that have these functions tied to the motor control jack.
RUN"filename": TC-8 searches for a specified program and runs it immediately.
PUT"filename": Same as SAVE "filename", except it is for use with system tapes.
GET: Same as LOAD, except it is for use with system tapes.
GET"filename": Same as LOAD "filename", except it is for use with system tapes.
GET? and GET?"filename": Same as LOAD? and LOAD?"filename", except it is for use with system tapes.
GETN and GETN"filename": Same as LOADN and LOADN"filename", except it is for use with system tapes.
OPEN: Required before cassette input or output of a data file can be attempted.
CLOSE: Required to end a cassette data file.
PRINT#: Allows numerical or string data to be output to a cassette file.
INPUT#: Allows numerical or string data to be input from a cassette file.

I haven't counted them, so I don't know about the "one load in a million bytes" claim, but my son, Anthony (age 11), loaded about 30 of his programs from his Radio Shack format tape to a new TC-8 format tape. He's run them all and found no bad loads.

Unlike the standard tape system, you can position your tape anywhere before the program you want and not have to look for a blank spot between programs. The TC-8 patiently waits for the program you want and then starts loading without getting confused by the portion of the previous program you just fed it.

Try that on your regular cassette system; you'll wear out the reset button. ■

ORDER NOW

To order your TC-8 kit, send your check or money order for $90.00 plus $3.50 postage and handling to JPC PRODUCTS CO., 12021 Paisano Ct., Albuquerque, NM 87112 (New Mexico residents add 4% sales tax). Credit card orders accepted by phone or mail. Personal checks will delay shipment. We will otherwise immediately ship you the TC-8 kit, the cabinet, the ribbon cable, the power adapter, an instruction manual, and a cassette containing the software.

✓190

JPC PRODUCTS CO.
Phone (505) 294-4623
12021 Paisano Ct.
Albuquerque, N.M. 87112

✓See List of Advertisers on page 386

ALIEN DEFENSE

MOD III with Sound

By Larry Ashmun

IT'S HERE!

"It's more than just another machine language arcade game."

IT'S AN EXPERIENCE

Super Fast

1 or 2 Player

MOD III ONLY

Tape version **$19.95**

Disk Version . . . . . **Coming Soon**

SEE OUR EXPANDED AD
IN 80 US JOURNAL
for more product information.

Larry Ashmun Is Back . . .
and Brings You . . .

SUPER VADERS

With Sound - Mod I & Mod III

©1981 Soft Sector Marketing, Inc.

The third in the evolution of the most popular space invaders games for Tandy machines. Now for 1 or 2 players, 10 levels of play.

Cassette (goes to disk) **$19.95**

Ask for upgrade information for TRS-Super Invaders or Invaders-Plus.

MOD III ROM COMMENTED

©1981 Soft Sector Marketing, Inc.

Only $22.50
+$2.50 Shipping & Handling

Not just a rehash of old information, but detailed comments on the ROMS in the latest machine from Tandy.

Now in its 3rd printing!
Over 150 Pages!

TAPE COPY 2

©1981 Soft Sector Marketing, Inc.

This program will load most any TRS-80 500 Baud system tape (standard) Mod I speed) and load it into memory and save it at either 500 or 1500 Baud on the Mod III. NO KNOWLEDGE OF MACHINE LANGUAGE NEEDED. Now it gives you a way to back up a machine language program that loads at the lower speed and makes cassette loading into your new Mod III a much faster and more reliable process. Works with Mod I* & Mod III.

Only **$14.95**

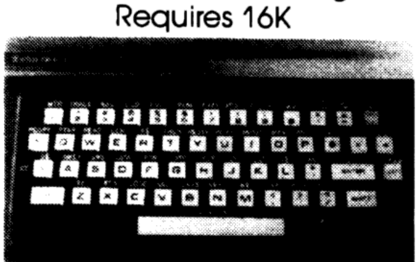
*Mod I loads and saves at [illegible] Baud only

MASTER CONTROL

For the Color Machine

©1981 Soft Sector Marketing, Inc.

Requires 16K



Ask about our other Color Machine programs!

FEATURES:

1. 50 preprogrammed command keys, standard and extended commands.
2. Direct control of motor, trace, and audio.
3. Relocatable machine code.
4. Automatic line numbering, starting point and increment alterable.
5. Programmable custom key.
6. Direct Run button.
7. Keyboard overlay for easy program use.
8. Easy entry of entire commands into the computer and display.
9. Full instruction manual.

Cassette **$24.95**

Outside Michigan
Order Line Only
800-521-6504

SSM SOFT SECTOR MARKETING,
INCORPORATED
6250 Middlebelt
Garden City, Michigan 48135

✔ 515

Questions &
Michigan Orders
313-425-4020

Metabasic—an end to the development of new languages.

A Macroprocessor For Basic—Part IV

J. Alan Olmstead
J. Olmstead Financial Engineering Systems
3813 West St., Moritz Lane
Phoenix, AZ 85023

Unless the Basic programmer is writing in native MetaBasic mode, the Basic compiler will always be an "emulator" compiler (an emulator is a program which reads from one language and translates into a similar language). I wrote my first assembly language emulator in 1967 and can state flatly that no emulator can function with 100% accuracy. I do not write in MetaBasic until the second state of program implementation, preferring to "sketch out" the outline of the program in fast and easy interpreter Basic. This is especially desirable for debugging tricky sections of logic. The interpreter may be relatively slow, but the problems are caught right at the time that the subroutine is being written and is, accordingly, fresh in the mind. That portion is then precompiled into MetaBasic and there are always details to fuss over and change, as much out of style as out of necessity. After a trial compilation and execution under assembly language, any portions of the program which are not exactly as you intended may be written in pure assembly language.

Programs should not be compiled for frivolous reasons. If the application is not time-dependent, is not in need of direct hardware controls for some operational reason, and is not subject to a concern for security, it should remain under interpreter Basic. Although not difficult, the technical environment involves so many details it's not reasonable for anyone, at any level of experience, to expect to realize quick and easy results.

Preparation

Once determined a program needs compilation, preparation is required for the programmer and the program.

The programmer should be well-trained or experienced to the extent that he understands the workings of every Basic command word he uses without reference to the Basic manual. Understanding does not mean mere memorization; it includes an understanding of the conditions under which the instruction will produce its particular kinds of errors and "unspecified results." The programmer should be totally familiar with the compiler manual. The programmer should also know how the computer hardware works, including what the registers do and how they do it.

The program to be compiled should itself be prepared, keeping in mind that in the compilation process the complex and general are broken down into the elemental and explicit. Multiple-command lines should be broken apart. An error message directing one's attention to a line containing a half-dozen to a dozen commands is nearly as worthless as no message at all.

Error-trapping subroutines are most commonly deficient in applications and systems software, but they are among the most important program elements. The purpose of error-trapping generally is to prevent the uncontrolled termination of program execution by attempting to correct or work around a condition for which the program or the operating system is not prepared. For example, when TRSDOS displays the Ready prompt, preceding the typed in response by a blank space causes the system to respond "What?" If an otherwise valid sequence of characters is typed in which is not a functional command word, the system attempts to locate a program file by that name. Failing, it displays the "Program Not Found" message. Although part of the normal functioning of the TRSDOS program, these are actually error traps.

The best example of an untrapped error is illustrated in For. . . Next loops which reference an unverified third element (see Fig. 1). The non-trapped code is sure to fail at one time or other, while the trapped code cannot fail—at least due to the LEN(Y$) as compared to the values possible for "X" due to "A". This kind of trapping may comprise ten percent or more of a program.

Error-trapping for I/O files is particularly important, and the methods differ depending upon the types of files and the types of access modes used for the files.

Most Basic programmers have never experienced a true program "blow up"; the closest they have experienced is a sudden and unexpected clearing of the screen and display of the Ready prompt in what is obviously a Boot-up system status. When a computer program blows up, the computer actually stops. To find out what went wrong requires the skills associated with the debug function, plus those of an experienced Assembly programmer.

Ample use of error-trapping procedures will assure the applications program will

"If the command returns to a zero, it means an error-free operation was performed since the last error.

always give a positive indicator of what problem has occurred. Unlike interpreter Basic, the MetaBasic compiled program will not stop program execution arbitrarily when a problem has been encountered. Instead, the error condition register is posted with the code number for the error; the error-trap register is checked to determine if an error-trap line number is active; if it is, the error exit takes place; if not, the operating system merely returns to the user's program after refusing to perform the requested function. The error code may be retrieved at any time by the command "A = ERR". If the command returns a zero, it means an error-free operation was performed since the last error. Every operating system function except ERR resets the error-trap register to zero before performing its requested function.

How the MetaBasic Compiler Works

Nearly every MetaBasic Compiler user will be initially interested in compiling already existing interpreter Basic programs, as opposed to writing new programs specifically for the compiled environment. Accordingly, the Precompiler will be as important as the Compiler itself.

Except for certain definition functions, the first word in a MetaBasic command is always a command word. Thus, the principal job of the Precompiler is to review the interpreter Basic program to ensure every command may be translated into MetaBasic format, and to do so if possible. If not, an error diagnostic is inserted conspicuously into the output MetaBasic program file.

The Precompiler is a group of three programs. The first program catalogs all referenced line numbers and makes certain all such lines actually exist. The first line of the program is always defined as a referenced line number, because it eventually will be, even if the program in its original interpreter Basic form does not reference it. Since Assembly does not use line numbers, all referenced line numbers are converted into name tags by inserting the letter L before the number. These appear as new lines in the form "L1000 EQU $", which is called an "equate" pseudo-operation command used by the assembler to identify a referenced address. From this point onward, the actual

Non-trapped Code	Trapped Code
1000 FOR X = 1 TO A STEP 1	1000 B = LEN(Y$)
1010 MID$(Y$,X,1) = "0"	1010 FOR X = 1 TO A STEP 1
1020 NEXTX	1020 IF X>B THEN 1050
	1030 MIS$(Y$,X,1) = "0"
	1040 NEXT X
	1050 . . .

Fig. 1

"Effective use of the Concur command is probably the most powerful component in Meta-Basic."

line number of a given line is of no importance.

Because the first line of the program is always the first line to be executed, the precomiler removes all non-executable lines of code (which would normally be ignored by the interpreter) and reinserts them above the first line. These include all dimension statements all data statements, and all declarations of names, both public in the present program and external in other programs. Immediately after the first line, the precompiler will insert all implied Open commands (screen, keyboard, printer), giving the default initialization values for those devices. A Print command will generate a run-time error if the printer is not an initialized ("opened") condition.

Concur

If the user wishes to break the present program into workable parts, or wants other program modules to be loaded into memory at the same time as the present program, he may insert "Concur" pseudo-commands as the first operations performed at the first line of the program. The commands permit naming up to six other program modules to be in memory concurrently with the present program. Concur is exactly like Chain except that control always returns to the user's program instead of being transferred to the newly loaded program module. Effective use of the Concur command is probably the most powerful component in MetaBasic.

MetaBasic is space inefficient on the disk file: This design decision was made deliberately for simplicity and maximum user information. Throughout the MetaBasic precompilation and compilation process, a single ASCII-format sequential file and workfile are used for all stages of program output. Error messages are embedded directly into the erroneous line, and it is recommended the user make ample use of both screen LISTs and printed listings.

In addition to error diagnostic insertions, the precompiler formats the Basic program into the beginning of Assembly field formats. After the line number, the first field is the tag field, the second is the opcode, the third is the operand field and the fourth, if present, is the comment field. The precompiler always begins the opcode field in the same place, whether a tag has been used or not. This greatly enhances readability but is not required for correct precompiler or compiler operation. Field structure is completely variable and there is no need for blank spaces between the operand elements.

Finally, the precompiler expands the interpreter Basic program by as much as two times the original number of lines. This is caused by the insertion of tag equates and reprogramming. There are many Basic interpreter commands which simply do not exist in MetaBasic; for example, an If. . . Then statement which contains the logical OR or AND functions may cause the expansion of as many as one dozen lines of emulating logic. In another example, numeric literals, quoted string literals and functions (VAL, ASC, ABS, etc.) may not appear in any command except a move-data command (SETBCD and SETSTR).

So that these do not have to be reprogrammed by hand, the precompiler removes them from the original command line, equates them to a working location, and substitutes the working location name back into the original command line. During compilation, the program grows to at least four times its original number of lines as illustrated by the move-data commands in Fig. 2. While the resulting Assembly program is a fraction of the size of its source counterpart, the source file can become huge. Some effective method of copying with large files must be provided.

The method is the Concur command, together with the source-level address linkage utility. Working with the MetaBasic source program file, the program may be unceremoniously chopped into halves, thirds, quarters, etc., and established as stand-alone files (PROG made into PROGA, PROGB, PROGC, PROGD, etc.). The resulting pieces of the original program are then run through the "external resolution generator" utility. Every data location and line name in each of the parts referenced by any one of the other parts will be automatically named "Public" in the part and "External" to the other parts which reference it. The result is that all parts communicate among each other as easily as though they were still one program file. The external reference generator may be run as often as needed, if changes are made.

The external reference generator utility also knows which program module is the first or main module. At the beginning of that module it will insert the Concur commands necessary to cause the Z-Monitor loader to load all subordinate modules as soon as the main module has been loaded.

After the first program of the precompiler has generated all the line number reference tags, it is a good idea to look at the program file and determine whether it should be subdivided into two or more program files at this point. Generally, if the program file is longer than 8-10 granules or 20K disk space, subdivision is strongly recommended. It is not necessary to do so, however, and the user may elect to continue on to the second of the three precompilation programs.

But Is It Legal?

The second precompiler group program analyses all command syntax except for dimensions and input/output operations of all kinds. The general rule for syntax analysis is, if the interpreter Basic form is not legal under the interpreter, it will be flagged and abandoned; if, however, it is legal under the interpreter and legal under MetaBasic in an altered form, it will be automatically reprogrammed to save manual labor. If it is legal under the interpreter but illegal under MetaBasic in any form, it will be flagged and abandoned.

The thrust during second-stage precompilation is what is legal under the interpreter but illegal in any form under MetaBasic. MetaBasic requires only one command (or one command and a comment) per numbered line. Error-trapping, already mentioned above, is one reason for this; another is that the logic of multi-command lines invariably becomes intermixed with the interpretation of If. . . Then statements. If. . . Then statements not only do not exist in that form in MetaBasic, there are no consistent rules for their interpretation among the various versions of Basic.

The second restriction, which will annoy many interpreter Basic programmers, is against complex functions and multi-function commands. For example, the precompiler will correctly handle "A = A + B*(C–D) /2". In truth, the author was reluctant to take on the responsibility for interpretation of fluid logic in addition to that for creating error-free Assembly code. Another example

MetaBasic		Assembly			
1000	SETSTR A$ = B$	1000	;SETSTR	A$ = B$	
		1010		CALL	SETSTR
		1020		DEFW	A$
		1030		DEFW	B$
1040	SETBCD A = VAL(A$)	1040	;SETBCD	A = VAL(A$)	
		1050		CALL	VAL
		1060		DEFW	A
		1070		DEFW	A$

Fig. 2

"A single test is made—greater-than for incrementing loops and less-than for decrementing loops."

of rejected logic, in complex functions, is "A$ = STR$(VAL(B$)/ABS(D))".

The second state precompiler will not annoy the user with mere violations of form. For example, if the user tests an embedded literal (not allowed), the precompiler will remove the literal into a generated workspace (see Fig. 3). Because If...Then statements have no direct counterpart in MetaBasic but due to their program frequency and importance, there is no variation, including logical OR and AND, which the precompiler will not automatically handle correctly so long as the user does not embed complex functions and computations.

One final restriction is common to all compilers. For...Next...Step commands must occur in perfectly matched pairs; the command Next by itself (as opposed to a named variable, such as "NEXT X") is not permitted. There are no practical restrictions on the number of For...Next loops which may be nested inside one another.

For...Next loops are another form of instruction which does not exist either in MetaBasic or in Assembly and must therefore be emulated. The variables will be initialized, the first loop will be executed, and the first test-after-incrementation will take place. The default condition for testing is increment. Decrement will not take place unless the Step argument is a negative literal (the compiler cannot test a value which will be known only at run-time). A single test is made—greater-than for incrementing loops and less-than for decrementing loops.

The Third Program

The third program handles all physical commands, including DIM, Data, Read, PEEK, POKE, Set, Reset, Restore and all input/output commands including Input, Print, LPRINT, Get, Put, LINEINPUT and PRINT#. MetaBasic dimension statements are structured closely to those of interpreter Basic; but their internal handling is very different. If the user has not explicitly dimensioned his data names, the precompiler does it for him with default values which must be checked for acceptability. Internal data is essentially handled as under the interpreter; it is formatted wherever it is encountered in the program, the precompiler ensuring that program logic jumps over the data areas. The address of each datum is entered into a table beginning with a table element pointer and ending with a table terminator word.

Both DIM and Data have interesting MetaBasic options. Since the program being precompiled may have been subdivided, and since data variables should not be di-

Basic	MetaBasic		
1000 IF A$ = "DATA" THEN 2000	1000	SETSTR	WK$002 = "DATA"
	1020	COMPSTR	A$ TO WK$002
			THEN . . .

Fig. 3

✓See List of Advertisers on page 386

"... MetaBasic or Z Monitor disk format is nothing like the TRSDOS disk format."

mensioned twice in one program, DIMs may be changed to External pseudo-operations by so indicating at the beginning of the job. Internal data, however, will be automatically set up in a new table within each program module. This presumes the programmer may wish to have program modules created for the sole purpose of organizing large volumes of data in kind of internalized indexed-sequential Read format, with the associated ability to restore to the beginning a part of the total data without having to restore it all the way back to the beginning of a possibly quite large table. When originally programming in MetaBasic, the programmer may designate which of 100 internal data tables a unit of data should be assigned to by appending the table number to the Data command, as in "DATA62 A,B,C,D," THIS ",E,1,2". This string of data would be assigned to table-name "DATA62", even if it were the only data table in the program. The corresponding Read and Restore commands would be "READ 62" and "RESTORE 62".

PEEK and POKE function as in interpreter Basic; their values are decimal.

The Set and Reset commands are exactly as under interpreter Basic, but they will be replaced by the "Draw Dot" and "Draw Blank" commands, respectively, which are part of the command set for business graphics.

Input/Output

MetaBasic recognizes five different standard I/O devices for the TRS-80 Model I: screen, keyboard, line printer, disk drives and serial I/O port (communications). Each has its own particular command words which may not be intermixed at the Basic or MetaBasic level (see Fig. 4).

Clearly, the majority of these commands cannot be written under interpreter Basic, and illustrate the paucity of available command words available to the Basic programmer given the range of peripheral equipment available to the TRS-80 Model I user today. It is important, however, that the user understand what will happen to the I/O commands in his interpreter Basic program when processed by the precompiler.

The Conversion

An easy way to remember the conversion is: *display* to the screen, *input* from the keyboard, *strobe* for a depressed key, *prints* to the line printer, and *read* and *write* from or to a disk file. PRINT@ commands generate a "CURSOR 1,p" command before printing. Unlike interpreter Basic, MetaBasic never adds an automatic carriage return at the end of the printed line. If the interpreter Basic line ends in a semi-colon, the precompiler will not generate the "SCROLL UP,1" command for the screen and the "LINE UP,1" command for the line printer. In both cases, printing null (PRINT"" and LPRINT"") and Print or LPRINT command without an argument will result in the appropriate line advance command. Where repeated Print and LPRINT commands have been used solely for vertical spacing, the single-advance commands which result may be consolidated into a single advance command with the number of lines desired as the argument, as in "LINE UP,5".

Sequential disk files are handled essentially the same as under interpreter Basic, except only one variable may send or receive data with each command. Accordingly, the command "LINEINPUT#1,A$,B$, C$" would result in three separate "READ LF1,[...]" commands.

Random access files are handled somewhat differently. There is actually no such thing as random access mode under MetaBasic. The Record I/O file format defines fixed-length records which may be written and read by sequential record number (beginning from one and continuing until end-of-file) in both sequential-access and random-access modes. There is no such thing as a Field statement, which is actually a clumsy adaptation from Cobol. To attempt to avoid manual reprogramming, the following conventions will be followed by the precompiler: The random-access file will be opened in both (read and write) ac-

Device/Mode	Command	Meaning
Screen (display)	SCROLL UP,n	Move lines up n lines
	SCROLL DOWN,n	Move lines down n lines
	SCROLL SET,n	Protect first n lines from scroll
	CURSOR [ON] [OFF]	Cursor visibility
	CURSOR 1,p	Position cursor at line, position
	DISPLAY A$	Print data on screen
Screen (graphics)	SCALE v,h	Ratio screen addresses to data
	PLOT v,h	Draw dot
	BLANK v,h	Draw non-dot
	DRAW LINE,v,h,v,h	Draw line from, to
	DRAW BOX,v,h,v,h	Draw rectangle within corners
	DRAW VBAR,w,v,h,v	Draw vertical bar of w-width
	DRAW HBAR,w,v,h,h	Draw horizontal bar of w-width
	FONT n,1,p	Change character size, lines, posns
	CHAR A$,v,h	Print data on screen
Keyboard	INPUT A$	Get data to a carriage-return
	STROBE A$	Check if key was struck
	ABORT AT nnnn	GOTO nnnn if CLEAR key struck during manual interrupt
Printer	PRINT A$	Print line on line printer
	LINE UP,n	Advance n-lines
	LINE DOWN,n	Reverse n-lines
	LINE EMS,n	Advance horizontally nth-inch
	LINE LEAD,n	Advance vertically nth-inch
	PROMPT [ON] [OFF]	Tell operator to change new page
	VMARGIN t,b	Set top/bottom page margins
	HMARGIN l,r	Set left/right page margins
	FONTSET [M] [A],A$	Manual/auto font name change
	PAGE n	Skip n-heads-of-form
	HEADER A$	Set repeating page header message
	FOOTER A$	Set repeating page footer message
	TABSET n,p	Set tabstop n at position p
	TAB n	Skip to tabstop n
Disk (variable)	READ LFn,A$	Read data from logical file n
	WRITE LFn,A$	Write data to logical file n
Disk (fixed)	READ FILE LFn,A$, REC = r	Read data from record r in logical file number n
	WRITE FILE LFn,A$, REC = r	Write data to record r in logical file number n
Serial I/O	BAUD nnnn	Set baud rate
	WORDLEN n	Set bits/word
	STOPBIT n	Set number of stop bits
	PARITY A$	Set parity odd, even, none
	FBYTE A$	Set automatic first byte
	LBYTE A$	Set automatic last byte
	XRACK	Send request for acknowledgement
	XACK	Transmit acknowledgement
	XNACK	Transmit negative acknowledgement
	XMIT A$	Transmit data
	RCV A$	Receive data
	TIMEOUT A	Set seconds to wait for answer

Fig. 4

cess modes. The Get command will read data into a precompiler-generated data location called "RA$n", where n is the logical file number assigned to the random-access file. The variable names in the Field command will be placed in a special lookup table. Whenever one of the Fielded variable names is referenced in a data move, the name will be changed a MID$ command which references RA$n by the subfield beginning number and length. Whenever an LSET command is encountered, the reverse will occur, and a Put command will again reference RA$n. The data name "DUMMY$" will be ignored in a Field command, but the position counters will be incremented. The precompiler will not attempt to handle more than two random access files per program segment.

It is equally important to note that the MetaBasic or ZMonitor disk format is nothing like the TRSDOS disk format. The directory is on track zero, not 17. Files have both names and numbers. Protect status is prevention against inadvertant destruction, not a futile exercise in pseudo-secrecy. A file's protect status is turned off by merely entering "PROTECT OFF". Most importantly, there are 64 different physical formats from which the user may choose in an effort to optimize space against speed. A disk record may contain as many as 2048 bytes or as few as 16. The Format command will optimize the physical placement of the records on a basis of so-many-per-track.

Whenever a sequential (variable length) file is declared, the disk surface remains formatted at ten sectors of 256 bytes per track. But when a record I/O (fixed-length) file is declared, the declaration must include the record length (RECL = nnn). Based upon this information, the named file's tracks are reformatted to physical records of the indicated length. Thereafter it is no longer necessary to block-up data records before writing to the file, nor to block-down to individual records after reading from the file. The numbered record is named and becomes available at record speeds.

This altered approach to disk I/O is not new; it is adapted from standard IBM disk I/O procedures. Accordingly, it makes possible a new high-speed lookup procedure called "indexed-sequential" organization.

This altered approach to disk organization also makes possible genuine high-speed disk-based sorting. The data itself is never sorted. Keyfiles are established and sorted, then the input files are rewritten to output file in an altered sequence based on the new order of the keys.

The output from the third stage of precompilation is supposed to be an error-free, pure MetaBasic source file. Since the file might have been written by hand, it may also be modified manually, including the insertion of the user's own machine code subroutines. MetaBasic source files may also be passed through the Assembly macroprocessor for this purpose. Every line of MetaBasic code is preceded by a colon at the first character position after the blank space which terminates the line number. After compilation, this will be changed into a semi-colon, which is the assembler's comment symbol. Assembly commands inserted by the user should not be preceded by a colon or a semicolon.■

Next month: More on the MetaBasic Compiler.

✓See List of Advertisers on page 386

With the help of Zen, build your own printer interface.

Hard and Soft Printware

I. R. Sinclair
89 Alexandra Road
Sible Hedingham
Halstead, Essex
England C09 3NP

By the time I had owned my TRS-80 for a few months, I had resolved that the first priority would have to be a printer. When your programs are short enough to list on the screen without scrolling, a printer seems an unimaginable luxury. But one of the joys of the TRS-80 lies in writing longer and longer programs using all the power of that big Basic. As a result, by the start of 1980, I was suffering severely from printout starvation, and only the use of a borrowed Teletype 33 kept me going.

About the same time, printer manufacturers must have noticed this problem, because there was a sudden flush of printers at $700 and less. Better still, they weren't thermal or electrostatic but real hard copy paper-markers. I suspect there was, in fact, only one mechanism with many manufacturers fitting electronics to it, but the result was certainly good for me. After some phone calls and a drive of a few hundred miles, I had a smaller bank balance and the feeling that I was, at last, winning the hard copy battle.

The Lowdown

I was right, but there was a long way to go. If you still want to jump on the hard copy bandwagon, this article will give you the complete lowdown of both hardware and software that will allow you to get into hard copy for only a few dollars more than the price of the printer.

It doesn't matter what type of printer you use, providing it has an RS232 interface built in. As far as I know, all the modern el cheapo printers use this type of interface, which lets us get away with simple attachment to the TRS-80 cassette port. The baud rate (the speed at which bits can be set from the computer to the printer) can be altered by setting the position of wire links inside the printer, and for this type of use the rate should be set at 300 baud.

Don't be tempted to set the rate higher in hopes of faster printing—the simple interface I am going to describe is reliable at 300 baud, but not at higher speeds. For higher speeds, a connection is needed from the printer to an input on the expansion interface connector of the TRS-80 so the printer can stop the computer when it has sent more characters than the printer can cope with. This type of connection is called a handshake, and my simple interface does not include this type.

Designing From Scratch

I had seen a simple interface described in a local user's group newsletter, but I doubted it would really work. I was right, so I set about designing my own interface from scratch.

The first thing I had to know was how large the signal from the cassette port would be. A cassette recorder does not need a large signal input, and too great a signal can cause distortion which leads to loading difficulties when the tapes are replayed. I bought only the keyboard unit of the TRS-80, and my cassette recorder is a sensitive one, so I had to cut down the signal strength by modifying the cassette output circuits. I used a voltmeter to measure the signal strength.

Step one was to get an output. Fortunately, on the TRS-80, it's not necessary to print a program to find out what happens at the cassette port. A simple program line was typed in and run:

```
10 OUT255,0:FOR N = 1TO1500:NEXT:OUT255,1:FORN =
   1TO1500:NEXT:GOTO10
```

This produced a voltage reading on the meter (attached to the cassette output socket) which varied from 0.46V to 0.48V and back again as the program ran. This is not a very large voltage swing for the input to an interface, though it's big enough for a cassette recorder. I started looking for ways to increase it without any further alterations to the TRS-80 itself. A quick look at the output circuitry showed that this could be done by altering the program to:

```
10 OUT255,2:FORN = 1TO1500:NEXT:OUT255,1:FORN = 1
   TO1500:NEXT:GOTO10
```

Simply by using a two in place of a one, the voltage swing became zero to 0.86 V, a lot greater and a much easier amount to deal with. I incorporated it into software for the printer.

The next step was to build the interface, an amplifier which would take the voltage swing of 0 to 0.86 volts at the cassette port and transform it to a voltage which (approximately) varied between −9V and +9V. These serial printers are tolerant animals, they don't appear to object if the pulses aren't strictly RS-232 standards. I was correct in assuming that a very simple amplifier would be good enough. Ten minutes on a solderless breadboard got me into hard copy, and a few hours work the next day (yes, I felt I had to make listings of all my programs in case it stopped working) resulted in a more polished version.

The Interface

The interface consists of a 741 operational amplifier IC, which draws its power from a rectifier bridge circuit which, in turn, is powered from the TRS-80 power pack. It uses only a small amount of power and constitutes no hazard to the TRS-80 supply. The unit is built into a small plastic box, and connects between the TRS-80 and its power

"Building a low-cost interface may give you a glow of achievement."

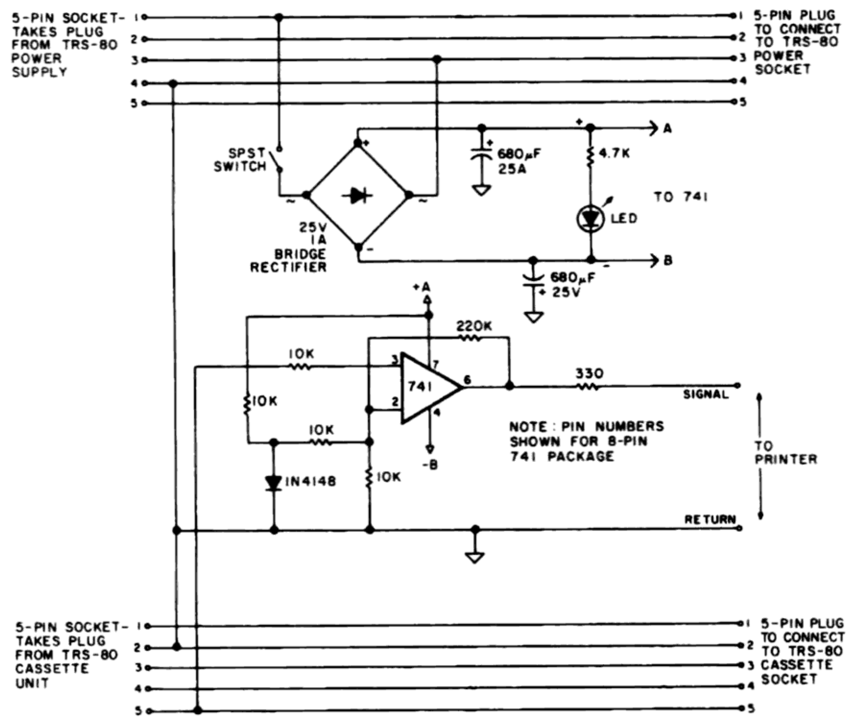


Fig. 1. This is the circuit for the interface unit. All resistors are ¼ W rating. The switch is optional, but if there is no online switch on your printer, it will prevent the printer from printing rubbish during a CSAVE.

and cassette plugs. The power plug from the TRS-80 supply goes into the interface, and the corresponding plug from the interface plugs into the TRS-80 power socket. The cassette lead is treated in the same way, so the interface can pick up power from the power cable, and signals from the cassette cable, but does not interfere with the operation of either. (There is no need to unplug the cassette unit, for example, when the printer is being used.)

The specific layout is not critical, and the circuit can be built in any way which you are accustomed to. I used Veroboard, but any type of matrix board which is designed for ICs is suitable. Make sure that the two smoothing electrolytic capacitors are correctly wired, because if the voltages are positioned wrong they can overheat and burst. The most tedious part of the whole effort is soldering the five-pin DIN plugs to fit into the TRS-80. Make sure before you start that the plugs will actually fit, because some plugs have elaborate covers which may prevent them from being correctly inserted.

For the benefit of you hardware pros, the circuit is a non-inverting amplifier. The signal from the cassette port is taken from pin 5 of the cassette socket through a 10K resistor to the + input on pin 3 of the 741. The − input on pin 2 is connected to a bias network, consisting of the diode and array of 10K resistors, and also to the feedback resistor of 220K. This feedback resistor also sets the amount of amplification. The output signal is taken from pin 6 of the 741 IC through a 330 ohm resistor. Power supplies of +9V and −9V are needed to operate the 741, and these supplies are obtained from the TRS-80 by connecting pins 1 and 3 of the power socket to a bridge rectifier. The output from the rectifier consists of the +9V and −9V lines, smoothed by 500µF capacitors. The cable to the printer does not need to use the expensive 25-pin Cannon-type plugs which so often appear on printer connections (using only two pins, it seems pointless to use a 25-pin plug). On my unit, I simply used a length of power cable, permanently attached at the interface. If you like plugs and sockets, a miniature two-pin unit is as good as any. The connection to the printer consists of the RS232 signal line and earth return only.

Check the circuit carefully, particularly the power supply section, because a fault here could damage the TRS-80 power supply. Testing will now have to wait until you have assembled the software!

The Software

Building a low-cost interface may give you a glow of achievement, but that will quickly dissipate if you have to spend $30 on the software to make it all work. Many serial printer programs exist, but none of them suited my needs, and so I developed my own, slowly and painfully. I wanted more than a straightforward routine, because the reason for buying a printer was so I could have listings of programs at various stages of development. (I hate keeping long rolls of paper, and I wanted a program which would list into neat pages of the same length as Radio Shack uses, 66 lines per page.)

Looking at a sample sheet which my printer had listed (when I bought it), I saw that the left margin was very small, too small for my filing methods, so another requirement for the routine was a left margin, preferably one which could be altered. The most important point, at first, was that the program should be fully relocatable, with no jumps to absolute addresses within the program. This way, I could use the same program at any address, or POKE it from a Basic program anywhere in memory, or as a string. I modified this requirement later!

My program specifications grew and grew. Having 66 lines per page implied that there would be a line counter, along with a reset method for the counter to ensure I could start a listing with the counter loaded to its full 66 lines. I wanted to use a command word for this rather than a POKE to memory. In addition, I wanted a keybounce delay (my '80 is an old one), and also some method of putting in a longer delay when I wanted it.

Fortunately, when it came time to write the program, I had the assistance of one of the best pieces of software I know of. It's an assembler-editor-debugger package for all Z-80 based machines, and its name is Zen. I don't know if Zen is available in the States; if it's not, I can pass on the name of the source in England.

Zen is simple to use—which means that you drop the manual quickly and concentrate on using it rather than trying to find out how to. As an assembler, it's versatile because entry is in free-format—you don't have to enter spaces of a set size, or even be careful about the number of letters in a name. Editing is very simple, just enter a new line, zap an old line or insert a new line. Assembler programs can even be combined from separate tapes, because the Zen line numbers are generated internally instead of being recorded on to the tape, as Radio Shack does. It also has a monitor section which lets you read in machine-code tapes, examine contents of memory, change codes, copy machine-code tapes, copy code from one set of memory locations to another, and even load code into memory as it is assembled by the assembler. This eliminates the tedious business of creating a tape and then reading it back in again. The

"The only problem is that there is no disassembler. . ."

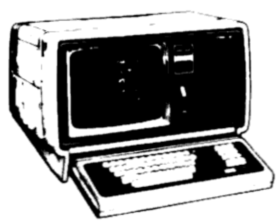
only problem is that there is no matching disassembler, but I am working on a program for reading in tapes produced by the Instant Software disassembler and producing from them a tape written to Zen standards.

The Zen listing is shown in Program Listing 1. There are no notes on the assembler listing itself, because I never have enough room on an assembler program for all the notes I want to make, and tapes take long enough to read without having all the extra characters loaded onto them. As I develop a program I keep all the original specifications. The amendment notes and listings result in my own home-made manual. This tells me more than a few notes on the program itself, and

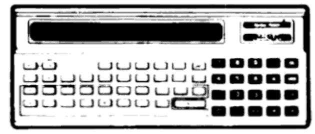
```
                      ;LPRINT ROUTINE WITH LHS MARGIN
                      ;66 LINES/PAGE AND 72 CH/LINE
                              ORG  7F11H
                      CHRSLN: EQU  4029H
                      LNPGE:  EQU  4028H
7F11 E5                       PUSH HL
7F12 21347F                   LD   HL,LPRT
7F15 222640                   LD   (4026H),HL
7F18 21F27F                   LD   HL,KBFIX
7F1B 221E40                   LD   (401EH),HL
7F1E 2ADE7F                   LD   HL,(CHRLN)
7F21 222940                   LD   (4029H),HL
7F24 21DF7F                   LD   HL,TOPPAG
7F27 227D41                   LD   (417DH),HL
7F2A 21E77F                   LD   HL, DELAY
7F2D 228341                   LD   (4183H),HL
7F30 E1                       POP  HL
7F31 C37200                   JP   0072H
7F34 F3               LPRT:   DI
7F35 79                       LD   A,C
7F36 F5                       PUSH AF
7F37 D5                       PUSH DE
7F38 FDE5                     PUSH IY
7F3A CD3300                   CALL 0033H
7F3D FDE1                     POP  IY
7F3F D1                       POP  DE
7F40 F1                       POP  AF
7F41 FE0D             ENDLN:  CP   0DH
7F43 2805                     JR   Z,RELOAD
7F45 FE20                     CP   20H
7F47 D8                       RET  C
7F48 181C                     JR   START
7F4A E5               RELOAD: PUSH HL
7F4B F5                       PUSH AF
7F4C 21DE7F                   LD   HL,CHRLN
7F4F 3A2940                   LD   A,(CHRSLN)
7F52 77                       LD   (HL),A
7F53 21DC7F                   LD   HL,LNCNT
7F56 35                       DEC  (HL)
7F57 200B                     JR   NZ,RUN
7F59 2A2840                   LD   HL,(LNPGE)
7F5C 22DC7F                   LD   (LNCNT),HL
7F5F 21DD7F                   LD   HL,ENDPAG
7F62 3606                     LD   (HL),06H
7F64 F1               RUN:    POP  AF
7F65 E1                       POP  HL
7F66 F5               START:  PUSH AF
7F67 E5                       PUSH HL
7F68 C5                       PUSH BC
7F69 0609             AGAIN:  LD   B,09H
7F6B 37                       SCF
7F6C F5                       PUSH AF
7F6D F5                       PUSH AF
7F6E 2101FC                   LD   HL,0FC01H
7F71 CD2102                   CALL 0221H
7F74 CD797F                   CALL DLY
7F77 1809                     JR   ROUND
7F79 21DE00           DLY:    LD   HL,00DEH
7F7C 2B               DLY1:   DEC  HL
7F7D 7C                       LD   A,H
7F7E B5                       OR   L
7F7F 20FB                     JR   NZ,DLY1
7F81 C9                       RET
7F82 F1               ROUND:  POP  AF
7F83 1F                       RRA
7F84 F5                       PUSH AF
7F85 3005                     JR   NC,OUT1
7F87 2102FC                   LD   HL,0FC02H
7F8A 1805                     JR   OUT2
7F8C C600             OUT1:   ADD  A,00H
7F8E 2101FC                   LD   HL,0FC01H
7F91 CD2102           OUT2:   CALL 0221H
7F94 CD797F                   CALL DLY
7F97 10E9                     DJNZ ROUND
7F99 2102FC                   LD   HL,0FC02H
7F9C CD2102                   CALL 0221H
7F9F CD797F                   CALL DLY
7FA2 F1               OUT3:   POP  AF
7FA3 F1                       POP  AF
7FA4 FE0D                     CP   0DH
7FA6 2022                     JR   NZ,TEST2
7FA8 21DD7F           TEST1:  LD   HL,ENDPAG
7FAB AF                       XOR  A
7FAC B6                       OR   (HL)
7FAD 2812                     JR   Z,BLNK
7FAF 35                       DEC  (HL)
7FB0 3E0D                     LD   A,0DH
7FB2 18B5                     JR   AGAIN
7FB4 C1               OUT4:   POP  BC
7FB5 21DE7F                   LD   HL,CHRLN
7FB8 35                       DEC  (HL)
7FB9 201D                     JR   NZ,OUT5
7FBB E1                       POP  HL
7FBC F1                       POP  AF
7FBD 3E0D                     LD   A,0DH
7FBF 1889                     JR   RELOAD
7FC1 21DB7F           BLNK:   LD   HL,LHSPCE
7FC4 3603                     LD   (HL),03H
7FC6 3E20                     LD   A,20H
7FC8 189F                     JR   AGAIN
7FCA 3ADB7F           TEST2:  LD   A,(LHSPCE)
7FCD B7                       OR   A
7FCE 28E4                     JR   Z,OUT4
7FD0 3D                       DEC  A
7FD1 32DB7F                   LD   (LHSPCE),A
7FD4 3E20                     LD   A,20H
7FD6 1891                     JR   AGAIN
7FD8 E1               OUT5:   POP  HL
7FD9 F1                       POP  AF
7FDA C9                       RET
7FDB 03               LHSPCE: DB   03H
7FDC 42               LNCNT:  DB   42H
7FDD 00               ENDPAG: DB   00H
7FDE 48               CHRLN:  DB   48H
7FDF E5               TOPPAG: PUSH HL
7FE0 21DC7F                   LD   HL,LNCNT
7FE3 3642                     LD   (HL),42H
7FE5 E1                       POP  HL
7FE6 C9                       RET
7FE7 F5               DELAY:  PUSH AF
7FE8 3AF67F                   LD   A,(KBFIX+4)
7FEB EE10                     XOR  10H
7FED 32F67F                   LD   (KBFIX+4),A
7FF0 F1                       POP  AF
7FF1 C9                       RET
                              ORG  7FF2H
7FF2 F5               KBFIX:  PUSH AF
7FF3 C5                       PUSH BC
7FF4 012000                   LD   BC,0020H
7FF7 CD6000                   CALL 0060H
7FFA C1                       POP  BC
7FFB F1                       POP  AF
7FFC C35804                   JP   0458H
                              END

AGAIN   7F69 BLNK    7FC1 CHRSLN  4029 CHRLN   7FDE
DLY     7F79 DLY1    7F7C DELAY   7FE7 ENDLN   7F41
ENDPAG  7FDD KBFIX   7FF2 LNPGE   4028 LPRT    7F34
LHSPCE  7FDB LNCNT   7FDC OUT1    7F8C OUT2    7F91
OUT3    7FA2 OUT4    7FB4 OUT5    7FD8 RELOAD  7F4A
RUN     7F64 ROUND   7F82 START   7F66 TEST1   7FA8
TEST2   7FCA TOPPAG  7FDF
```

Program Listing 1. This is the assembler listing. If desired, the object code can be typed in using T-Bug, or the program reassembled using EDTASM. For complete relocatability, the DLY subroutine and the storage addresses should be located in low RAM, such as 405CH to 407FH (which is used only during the switchon sequence).

Here's a REAL PLUS

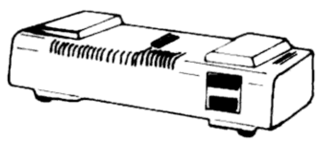


Model II 64K $3300

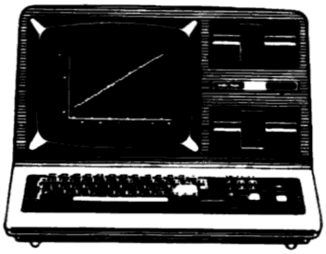


Pocket Computer $189
P. C. Printer $130

Color Computer 4K $310
w/16K Ext. Basic $459



32K Exp. Int. $334



Model III 16K $839
2 DR + RS232C $2100

LOOK Here are just a few of our fine offers . . . call toll-free for full information.

COMPUTERS	
Model II 64K	$3300
Model III 4K LEV I	599
MODEL III 16K	839
MODEL III 32K	945.50
*MODEL III 32K	881.50
MODEL III 48K	1052
*MODEL III 48K	924
Model III 48K 2 Disc & RS232 c	2100
Color Computer 4K	310
Color Computer 16K	416.50
*Color Computer 16K	352.50
Color Computer 16K w/extended basic	459
Pocket Computer	189
VIDEOTEX	310
APPLE 48K only	1279
ATARI 800 32K	789

*Computer Plus New Equipment.
180 Day Extended Warranty

PERIPHERALS	
Expansion Interface OK	$249
Expansion Interface 16K	355.50
*Expansion Interface 16K	291.50
Expansion Interface 32K	462
*Expansion Interface 32K	334
16K RAM N.E.C. 200 N.S. chips	25
MODEMS	
Lynx Direct Connect MI/MIII	269
Auto Ans./Dial	
Telephone Interface II	169
R.S. Modem I D.C.	130
Atari 830 Modem	159
Atari 850 Interface	183
PRINTERS	
Daisy Wheel II	1695
Line Printer VI	999
Epson MX80	499
Epson MX80 FT	599
Epson MX100	799
Line Printer VII	315
Line Printer V	1610
Microline 80	345
Microline 82A	515
Microline 83A	799
Pocket Computer Printer	130
DISK DRIVES	
R.S. Model III 1ST-Drive	712
TEAC 40 Track MI	329
R.S. 1 Drive Exp MII	999
R.S. 2 Drive Exp MII	1518
R.S. 3 Drive Exp MII	2040
Seimans 8" MII	799
Atari 810 Disk	499
SOFTWARE	
R.S. Software 10% off list	
Atari Software 10% off list	
Newdos + (40) track MI	88
Newdos 80 MIII	149
ST80III	149
ETC.	
Verbatum 5" Double Density	32
Verbatum 8" Data Life	49.95
Ctr-80A recorder	52
C. C Joysticks	22

PLUS real back-up warrantees —

Factory warrantees on Apple and Atari equipment. Other equipment carries manufacturer's warranty or Computer Plus 180 day extended warranty. Combined warrantees carry Computer Plus 180 day warranty or original manufacturer's warranty.

DEALER INQUIRIES ARE INVITED

Prices subject to change without notice.
Not responsible for typographical errors.

TRS-80 is a registered trademark of Tandy Corp.

call TOLL FREE 1-800-343-8124

computer plus

Dept. D
245A Great Road
Littleton, MA 01460
617-486-3193

Write for your free catalog

✓130

"This tells me more than a few notes on the program itself, and doesn't use memory."

doesn't use memory. The program is arranged to give a baud rate of 300 for the reasons described earlier, but for anyone who uses 110 baud machines, one single alteration will give the lower rate.

The program has its origin at 7F11H, 32529D, and two of the printer control block addresses are used for storage, and another two bytes for the address of the new routine. Location 4028H(16524D) is used to store the number of lines per page. This is the number which is placed in this address when the TRS-80 is switched on, so no addition has to be made here. Location 4029H is used in this program to store the number of characters per line, and POKEing a new value into this address will alter the number of characters per line which your printer will deliver.

Four addresses in the program space itself are used also for storage, and these can be POKEd to change the quantities such as the left margin space and the gap between pages. As set up, the program prints 72 characters per line, 66 lines per page, four spaces on the left margin, and six blank lines between pages. A complete description of the program follows so anyone who wants to build on it for his own purposes can do so.

The first section of the program from 7F11H to 7F2FH loads up the RAM locations. Address 4026, with 4027, contains the address of the printer routine. When the TRS-80 is switched on, the operating system loads the address 058DH into these locations, which is the start of the ROM routine for a parallel printer. Our print routine starts at 7F34H, so that this is the address which must be loaded to these locations. This way, we can use the commands LLIST and LPRINT rather than having to use new command words. There is a similar readdressing at 401EH,401FH. These addresses hold the address of the screen-print routine, normally 0458H, now directed to 7FF2H. This lets me put a delay into the screen-print routine, and serves two very useful purposes. One is that it debounces my keyboard, and the other is that it allows me to incorporate a slow-listing facility. The number of characters per line is loaded into 4029H, no load is done into 4028H because the regular 66 lines per page suits me. If you want a different number, add a couple of steps to accomplish this.

When a disk system is not in use several commands such as Field,Get,Put and about 25 others will return with L3 Error if you use them in Basic. According to the manual, this is because they are disk commands which work only when a disk operating system is loaded. In fact, each of these commands causes the operating system to look in RAM for the address of a routine, and if you put in your own routine, and POKE its address to the correct place, you can make use of any of these commands.

In this program, I have used Field to reset the lines-per-page count, so the printer will print a full 66 lines before it creates the six blank lines at the end of a page. The address of a page-reset routine is loaded into 417DH, the reference address for the Field command. Similarly, I have used the command Put to change the delay time by loading the address of a delay-change routine into 4183H, the address for the Put command. By typing Put and entering from Basic the TRS-80 screen-print rate is slowed so the listing can be watched on the screen. This, combined with the TRON command, is a considerable aid to debugging a Basic program, because with these two I can watch a program execute in slow motion.

Typing Put for a second time and entering takes the additional delay out again, and screen operations proceed normally. This delay can also be placed in the keyboard routine, but there are advantages to using the screen routine. One is that the delay acts only when the screen is accessed, as it always is when a character is being entered from the keyboard. A delay in the keyboard routine, unless it is a rather complicated delay program like the Radio Shack KBFIX, slows down all program operations because the keyboard is scanned continually as a program progresses. This, of course, is why the Break key can be used. If you disable the keyboard you lose all control over a program.

The loading section ends with a jump back to Basic at 0072H, and the printer routine itself starts at 7F34H. The DI command is used so the routine cannot be interrupted in mid-character by the Break or any other key. The printer routines of the TRS-80 place the character to be printed in the C register of the Z80, so the next step is to transfer this to the accumulator so we can operate on it. The section of the program from 7F36H to 7F40H saves the registers and calls the screen-print routine, and whatever is sent to the serial printer is also echoed onto the screen. This is a useful feature because most dot-matrix printers conceal what they are printing until several linefeeds have taken place. If you want to be able to cut this feature in and out, use one of the disk control words to POKE 00 into locations 7F3AH and 7F3BH, and another to POKE back the bytes which are shown in these locations.

At ENDLN the byte in the accumulator is checked to see if it is a carriage return (0DH), in which case the program branches to reload to reset the characters-per-line count and to decrement the number-of-lines

count in location LNCNT. If at ENDLN, the byte in the accumulator is not a carriage return and is less than the space byte (20H), the program returns to ROM for the next character, because the control characters other than carriage return are not used in the program (the dot-matrix printers can be set to line-feed on carriage return, so the additional complication of a line-feed is not needed). For any other character, the program then jumps to Start.

At Start, the registers are saved on the stack, and register B is loaded with a bit count of nine (one byte plus a space). A mark bit is sent out (my printer interprets a mark as a 0 and a space as 1) by loading FC01 to the HL pair, and calling the TRS-80 subroutine at 0221H. Working on the results of the cassette port output voltage mentioned earlier, the mark byte is 01 and the space 10. To establish the correct baud rate, a delay must follow, and this is achieved in the subroutine DLY. The byte 00DEH, which is loaded into HL at the start of this subroutine, determines the baud rate. So if you want 110 baud, a larger number must be loaded here—try 0267H.

At the label Round, the bits which make up the character are sent out one by one. The RRA command shifts the contents of the accumulator one place to the right, so the lowest-order bit lands in the carry position. If the bit in the carry position is 0, C is reset and the program jumps to OUT1 to send out a mark bit. If the bit in the carry position is a 1 then HL is loaded with FC02H, the space bit, and this causes the OUT2 routine to send out a space. Each of these routines are followed by the DLY subroutine to keep the baud rate correct. Yet, at 300 baud the timing does not have to be exact. The DJNZ command at 7F97H keeps the rotate and output routine going until the B-register has been decremented to zero in the usual action of the DJNZ command. A final delay follows, and the program jumps to OUT3, which is the end of the part of the routine that outputs the character to the printer.

At OUT3 the AF register is POPped to recover the original unrotated byte and this is compared with the carriage return byte, 0DH. If the byte which has just been sent to the printer was 0D, then the program skips to Test2 otherwise at Test1, the byte from ENDPAG is loaded in which clears the accumulator. Now if the lines-per-page count reached zero earlier in the program (at Reload), there will be a six in ENDPAG giving the OR (HL) step a positive result. The program will continue by decrementing ENDPAG by loading a carriage return character into the accumulator and jumping back to output this to the printer. This will continue each time this place is reached un-

✓See List of Advertisers on page 386

"My printer chatters away happily, and never has missed a character."

til the ENDPAG location is zero.

If the lines/pages count had not reached zero, the OR (HL) step gives a zero answer, and the program skips to location BLNK where LHSPCE is loaded in with the number three. The accumulator is loaded with the ASCII character 20H, and the printing routine is repeated, sending a space to the printer.

When the character in the accumulator was not a carriage return (and this includes the space which is loaded at BLNK), Test2 is used. This checks the byte in LHSPCE to see if more spaces need to be output to the printer, otherwise a retest is sounded by a jump to OUT4. At OUT4, the number of characters per line is decremented (note that the blank spaces at the left side are not counted) and if the count has reached zero, the accumulator is loaded with a carriage return and recycled to the printer. If the byte is not at the end of a line, the next stage in the countdown is OUT5, where the registers are restored to the original values, and the routine returns for the next character.

The rather odd jump sequences in this program so far allow it to be relocatable.

The bytes from 7FDBH to 7FDEH are storage bytes for the left-space, lines per page, end-of-page spaces, and characters per line respectively. These are fixed addresses which also make the program not completely relocatable, but there is no reason (apart from convenience) for having them in the high memory portion of the program. For example, they can be loaded into the top end of the cassette buffer memory or into disused disk command RAM; either way would make this part relocatable. At TOPPAG, a short routine restores the number of lines per page which is called by the Field command.

Similarly, at 7FE7H, another short routine calls a delay change routine, by taking the byte from 7FF6H, and X-ORing it with 10H. If the delay byte was the 00 set at the start of the program it is changed to 10 by this action. But if the byte was 10H, it is changed back to 00 by XOR. This routine is called by the Put command, and like the TOPPAG routine, it uses absolute addresses. This routine, like the TOPPAG, can be invoked from a running Basic program so screen scrolling can be controlled from within the program itself. Finally at 7FF2H, the video delay consists of saving registers, loading the BC pair with the delay bytes, calling the TRS-80 delay routine at 0060H, restoring registers, and then jumping to the screen routine in ROM at 0458H.

Now how did it live up to specification? It was not as easy to relocate as I had wanted, but that can be sorted out if needed. I had not realized at the start how easy it would be to reassemble at different addresses, thanks to the Zen assembler. In any case, full relocatability is not difficult to achieve, either by using low RAM memory for all fixed locations, or by loading registers through immediate loading. The real snag is the delay subroutine. This can be located in low RAM, or simply divided so there is a delay routine in each position where it is needed. However, I have no need for full relocatability, and the program performs very well indeed. My listings now have left margins, so they can be punched for filing, and they cut neatly into page. My printer chatters away happily, and has never missed a character. What more could I ask? ■

See List of Advertisers on page 386

Add a synthetic keyboard to your TRS-80.

Auto-Key

Mike Ghan
2210 108th Ave. S.E.
Bellevue, WA 98004

Ever need to input a complex mathematical expression during a Basic program run? How would you like to load machine-language files during execution without disturbing the video display or variables?

Auto-key adds a synthetic keyboard to your TRS-80. By synthetic, I mean a utility that uses a string of characters in place of the normal keyboard input routine.

Auto-key is fast and easy to use; just define a string in Basic and execute a USR command. You can enter any character that can be entered from the keyboard, plus Basic internal codes. For example, 185 is the code for CLOAD. If you enter CHR$(185) into your string, Auto-key interprets it as CLOAD.

Further, Auto-key saves 2.5K bytes of disk space over my previous method of auto power up.

Compatibility is no problem. You can use Auto-key with Model I 4K Level II through 48K disk systems. I have not had the opportunity to use Auto-key with a Model III, but I foresee no problems.

How Auto-key Works

Auto-key recognizes three control codes. Control code 1 disables the video driver. This allows you to work without disturbing the video display. Auto-key is initialized with the video driver disabled.

Control code 2 enables the normal video driver.

Control code 3 terminates Auto-key and restores all drivers to their original condition.

Auto-key's Assembly code is shown in Program Listing 1. Lines 170–220 process the USR parameter to point to the string location. This allows you to use a simple command: X = USR0 (VARPTR(A$)), where A$ contains the target string to be processed. The VARPTR function returns the string pointer, *not* the string address. Line 230 is the Assembly Language entry point and loads register HL with your target string address and Calls MACHMC. Line 240 saves the string address to be accessed later.

Lines 250–280 save the video and keyboard driver addresses so they can be restored. This ensures that other drivers, such as a key debounce routine, that may be active before Auto-key is run will be restored as well. These original drivers are also linked to Auto-key.

Lines 290–330 load Auto-key's address into the video and keyboard Device Control Blocks (DCB), and return to the calling program. Auto-key is now initialized and ready for execution.

"How would you like to load machine-language files during execution without disturbing the video display or variables?"

Lines 390–410 check for the up arrow key and terminate Auto-key if it is pressed.

Lines 420–440 check to see if the cursor is on. If it is not on Auto-key exits via the normal keyboard driver, allowing INKEY$ to remain functional.

Lines 450–480 bump the string pointer and load register A with the next character to be passed on.

Lines 490–540 test for control code 1, the code to turn off the video. If a 1 is found, the video off driver is loaded into the video DCB; register A is cleared; and a return sends you back to the caller.

Lines 550–620 check for a 3, the code for the end of the target string. If a 3 is found, all the original drivers are restored into their DCBs and Auto-key is

unlinked.

Lines 630–680 test for a 2, the code for video on. If a 2 is not found the character is assumed to be valid and is returned to the keyboard caller. If a 2 is found, the original video driver address is restored to its DCB.

Lines 720–780 are the video off routine. They check for a 14 (cursor on) or a 15 (cursor off). By replacing all other codes with a zero, this turns the video driver off. Lines 780 and 790 store the original driver addresses and serve as links for Auto-key.

One idea I would like to pass on is my use of jump (JP) vectors. If you look at RAM starting at 4000H, you will see a series of JP instructions. This allows various routine codes to be altered or updated. Since a change also changes the entry point for a given routine, any other routine which calls it would have to alter its call instruction. You need only change the jump vector to match the new entry point and everything goes just fine.

I use this concept a little differently. My jump vectors start in high memory and work down. Thus, I can move a routine or change it and all I need to do is change the jump vector to maintain compatibility with my many programs that access that particular routine.

Alterations and Enhancements

As written, Auto-key is not relocatable, but you could change that with a much longer routine. Enter the source code into your assembler and change the origin statement in line 110 to match your memory size or requirements. The 119 bytes of code for Auto-key can be contained in Basic data statements and POKEd into memory. Program Listing 2 uses this technique.

A program called Progdata (see *80 Microcomputing*, May 1980, page 126) will make the task of defining these data statements much easier. Also you can add the code for Auto-key to an existing machine-code program allowing easier loading and using less storage space.

Additional functions can be added to Auto-key by testing for another control code. For example, a CP 04 could be used to access a machine language routine. The code 04 would be followed by two bytes containing the entry point to the particular routine. By loading register HL with these two bytes and executing a JP (HL) instruction, you could have a vectored USR command. This would be very useful in Level II Basic which has only one USR command. Don't forget to clear register A upon return. I'll leave the loose ends for you to tie up. A little food for thought.

Typical Applications

Program Listing 2 contains a 16K Level II Basic program that inputs mathematical formulas directly during program execution. After loading the machine code in lines 1000–1080, the program prompts you with "Z = ?" in the line 120. Respond with a mathematical expression such as " 2*3 + 4*SQR(6)/6 ".

Line 130 adds the following:

- CHR$(2) turns on the video.
- "10 Z = ", the program line number and "Z = ".
- The formula you input, Q$.
- ":?Z", to print the value of Z.
- CHR$(13), the code for a carriage return. This enters the new line.
- The command RUN and CHR$(3), which runs the program with the new line 10 added. CHR$(3) is the end of

```
                  00100 ;         AUTO-KEY  by MIKE GHAN
F600              00110           ORG       0F600H
F600              00120 ZSTART    EQU       $
                  00130
                  00140 ;THE FOLLOWING BLOCK INITIALIZES DEVICE CONTROL BLOCKS FOR BOTH
                  00150 ;THE VIDEO AND KEYBOARD DRIVERS. RETURNS TO BASIC OR A CALLING
                  00160 ;MACHINE LANGUAGE ROUTINE
F600 CD7F0A       00170 USRMAC    CALL      0A7FH               ;GET USR PARAMETER
F603 23           00180           INC       HL                  ;POINTS TO STRING POINTER
F604 5E           00190           LD        E,(HL)              ;GET LSB OF STRING ADDRESS
F605 23           00200           INC       HL
F606 56           00210           LD        D,(HL)              ;GET MSB
F607 EB           00220           EX        DE,HL               ;SWAP BACK INTO HL
F608 2B           00230 MACHMC    DEC       HL                  ;MACHINE LANGUAGE ENTERY POINT
F609 2225F6       00240           LD        (MACPNT),HL         ;SAVE STRING POINTER
F60C 2A1E40       00250           LD        HL,(401EH)          ;GET VIDEO DRIVER ADDRESS
F60F 2272F6       00260           LD        (VIDOUT+1),HL       ;SAVE IT TO RESTORE
F612 2A1640       00270           LD        HL,(4016H)          ;GET KEYBOARD DRIVER ADDRESS
F615 2275F6       00280           LD        (KEYOUT+1),HL       ;SAVE IT ALSO
F618 2166F6       00290           LD        HL,NOVID            ;NEW VIDEO DRIVER
F61B 221E40       00300           LD        (401EH),HL          ;PUT INTO DEVICE CONTROL BLOCK
F61E 2127F6       00310           LD        HL,MACRO            ;NEW KEYBOARD DRIVER
F621 221640       00320           LD        (4016H),HL          ;PUT INTO DCB
F624 C9           00330           RET                           ;RETURN TO CALLER
                  00340
F625 0000         00350 MACPNT    DEFW      00H                 ;POINTS TO CHARACTER TO PASS
                  00360
                  00370 ;THIS IS THE ACTUAL AUTO-KEY DRIVER, IT CHECKS FOR VALID CONTROL
                  00380 ;CHARCTERS ( 01, 02, OR 03 ) AND GETS CHARACTERS IN THE STRING.
F627 3A4038       00390 MACRO     LD        A,(3840H)           ;TEST FOR "UP ARROW"
F62A FE08         00400           CP        8H
F62C 281E         00410           JR        Z,MACOUT            ;CANCEL AUTO-KEY
F62E 3A2240       00420           LD        A,(4022H)           ;GET CURSOR CHARACTER
F631 B7           00430           OR        A                   ;IF CURSOR IT NOT ON,
F632 2840         00440           JR        Z,KEYOUT            ;   EXIT VIA NORMAL KEY DRIVER
F634 2A25F6       00450           LD        HL,(MACPNT)         ;GET CHARACTER POINTER
F637 23           00460           INC       HL                  ;BUMP TO NEXT ONE
F638 2225F6       00470           LD        (MACPNT),HL         ;SAVE IT FOR NEXT TIME
F63B 7E           00480           LD        A,(HL)              ;GET THE CHARACTER
F63C FE01         00490           CP        01H                 ;TEST FOR "VIDEO OFF"
F63E 2008         00500           JR        NZ,MACON1           ;GOTO MACON1 IF NOT 01
F640 2166F6       00510           LD        HL,NOVID            ;VIDEO OFF DRIVER
F643 221E40       00520           LD        (401EH),HL          ;PUT IN DCB
F646 AF           00530           XOR       A                   ;RETURN ZERO (NOTHING)
F647 C9           00540           RET
F648 FE03         00550 MACON1    CP        03                  ;END AUTO-KEY?
F64A 200F         00560           JR        NZ,MACON2           ;GOTO MACON2 IF NOT
F64C 2A75F6       00570 MACOUT    LD        HL,(KEYOUT+1)       ;RESTORE ORIGINAL KEYBOARD
F64F 221640       00580           LD        (4016H),HL          ;   DRIVER TO DCB
F652 2A72F6       00590           LD        HL,(VIDOUT+1)       ;RESTORE ORIGINAL VIDEO
F655 221E40       00600           LD        (401EH),HL          ;   DRIVER TO DCB
F658 3E0D         00610           LD        A,0DH               ;RETURN A 'ENTER'
F65A C9           00620           RET                           ;ROUTINE NO LONGER LINKED
F65B FE02         00630 MACON2    CP        02H                 ;VIDEO ON?
F65D C0           00640           RET       NZ                  ;NO, RETURN VALID CHARACTER
F65E 2A72F6       00650           LD        HL,(VIDOUT+1)       ;ORIGINAL VIDEO DRIVER
F661 221E40       00660           LD        (401EH),HL          ;RESTORE DCB
F664 AF           00670           XOR       A                   ;RETURN ZERO
F665 C9           00680           RET
                  00690
                  00700 ;THE FOLLOWING ROUTINE TURNS VIDEO OFF BY SCREENING ALL
                  00710 ;CHARACTERS EXCEPT CURSOR ON ( 14 ) AND CURSOR OFF ( 15 ).
F666 79           00720 NOVID     LD        A,C                 ;REGISTER C HOLDS CHARACTER
F667 FE0E         00730           CP        14D                 ;CURSOR ON CODE?
F669 2806         00740           JR        Z,VIDOUT            ;YES, PASS IT ON
F66B FE0F         00750           CP        15D                 ;CURSOR OFF CODE?
F66D 2802         00760           JR        Z,VIDOUT            ;YES, PASS IT ON
F66F 0E00         00770           LD        C,0H                ;PASS A ZERO (NOTHING)
F671 C30000       00780 VIDOUT    JP        0000H               ;REPLACE 0000H WITH ORIGINAL
F674 C30000       00790 KEYOUT    JP        0000H               ;    DRIVER ADDRESS AT START
F677              00800 ENDMAC    EQU       $
402D              00810           END       402DH               ;CHANGE TO 1A19H FOR LEVEL II
```

Program Listing 1. Auto-key

string code for Auto-key that returns Enter for the last command.

Line 140 executes a USR call to enable Auto-key. The command Stop does two things. First, you can't enter a program line during execution, and second you need to have the cursor on to activate Auto-key. The cursor needs to be on because Basic is continuously scanning the keyboard looking for a Break key or setting INKEY$ values. Without checking for the cursor, Auto-key would be returning the first part of your string before Basic found the stop statement. Try running the program, inputting various expressions, and then hit Break and list the program. The last expression you enter will be in line 10. "BREAK IN 140" is never displayed because Auto-key is initialized with the video driver disabled. Disk Basic users will have to reassemble Auto-key into higher memory and use the DEF USR1 = &Hxxxx statement to define the user entry point.

Program Listing 3 contains a Disk Basic program that lists the disk directory, returns to Disk Basic and resumes execution. Since the program operates in a similar manner to Program Listing 2, I will explain the target string (DIR$) only.

- CHR$(133) is the Basic internal code for CMD.
- The CHR$(34)s put quotes around S. So far we have CMD"S".
- CHR$(13) and CHR$(2) add Enter and turn on the video, respectively.
- "DIR (A, I, S,)" and CHR$(13) execute the directory read with file allocation, invisible files and system files.
- CHR$(01) turns the video off again.
- "BASICR*" and CHR$(13) re-enter Disk Basic with the program and all variables intact.
- CHR$(179) is the Basic internal code for CONT.
- CHR$(3) terminates Auto-key and returns an Enter. The Basic program now resumes execution at the 140.

Program Listing 4 is an example of auto power up to Disk Basic. This routine is part of a larger program called Control, that contains my upper/lowercase driver, joystick driver, screen print driver, home interface controller and Auto-key.

TRSDOS allows one power up command via the Auto command. Auto loads Control on power up with the execution address as line 110 (COMPRC). From there Auto-key takes over.

Line 110 calls a routine that initializes my DCBs to point to various drivers so that when Auto-key is finished these pointers will be restored. (Line 110 is only necessary if you use driver routines of your own).

Next, line 120 loads register HL with the string address called MACBUF. Line 130 calls Auto-key's machine-code entry point (MACHMAC in Program Listing 1).

Lines 140–150 perform a very neat trick. Memory location 4049H contains the TRSDOS top of memory pointer. If you alter this pointer, your TRS-80 thinks it only has memory up to that number. Since Disk Basic uses the top 64 bytes of memory for initialization, any code in the top 64 bytes of high memory will be clobbered. By changing the top of memory pointer to an address lower than any of your routines, you can avoid this problem. You can answer the memory size question with Enter and both Basic and TRSDOS will leave this high memory alone.

Line 160 jumps back to TRSDOS where it turns on the cursor and calls the keyboard driver. Auto-key intercepts and passes back the string in lines 170–270. Line 280 terminates Auto-key's control.

With some further effort, you can construct more complex programs during run time, using Auto-key. ■

```
10  GOTO1000: REM MEMORY SIZE = 32628
100 CLEAR300:DEFINT A-Z
110 POKE 16526,136:POKE 16527, 127 : REM  AUTO-KEY ENTRY POINT
120 INPUT"Z= ";Q$
130 Q$=CHR$(2)+"10 Z="+Q$+":?Z"+CHR$(13)+"RUN"+CHR$(3)
140 X=USR(VARPTR(Q$)):STOP
1000 FOR Z8= 32648TO 32767:READ Z9:POKEZ8,Z9:NEXT Z8:GOTO100
1010 DATA 205,127,10,35,94,35,86,235,43,34,173,127,42,30,64
1020 DATA 34,250,127,42,22,64,34,253,127,33,238,127,34,30
1030 DATA 64,33,175,127,34,22,64,201,0,0,58,64,56,254,8,40
1040 DATA 30,58,34,64,183,40,64,42,173,127,35,34,173,127,126
1050 DATA 254,1,32,8,33,238,127,34,30,64,175,201,254,3,32
1060 DATA 15,42,253,127,34,22,64,42,250,127,34,30,64,62,13
1070 DATA 201,254,2,192,42,250,127,34,30,64,175,201,121,254
1080 DATA 14,40,6,254,15,40,2,14,0,195,0,0,195,0,0,0
```

Program Listing 2. This Basic program will list mathematical formulas during a program's execution.

```
50  REM CHR$(133)="CMD", CHR$(179)="CONT"
100 CLS:CLEAR500:DEFINTA-Z
110 DIR$=CHR$(133)+CHR$(34)+"S"+CHR$(34)+CHR$(13)+CHR$(2)+"DIR (
A,I,S)"+CHR$(13)+CHR$(01)+"BASICR *"+CHR$(13)+CHR$(179)+CHR$(3)
120 PRINT"ACCESSING DIRECTORY"
130 DEFUSR1=&HFFF3:X=USR1(VARPTR(DIR$)):STOP:REM FFF3H IS JUMP V
ECTOR FOR AUTOKEY
140 PRINT"DIRECTORY READ":END
```

Program Listing 3. This Disk Basic program lists the directory and resumes programming.

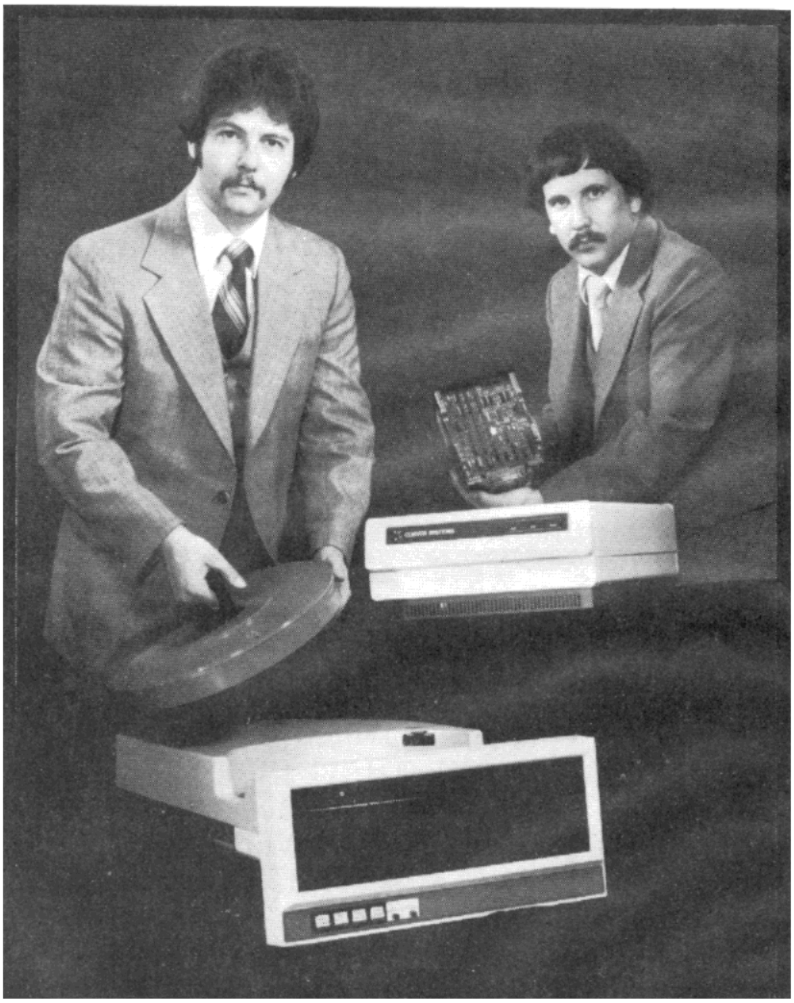
```
                00100 ;                    C O N T R O L  V 6.2   2/23/81
UNDEFINED SYMBOL
0000 CD0000     00110 COMPRC  CALL    SETUP          ;SET DCB,s. OPTIONAL
0003 211200     00120         LD      HL,MACBUF      ;STRING ADDRESS
UNDEFINED SYMBOL
0006 CD0000     00130         CALL    MACHMC         ;AUTO-KEY ENTRY POINT
0009 2100EF     00140         LD      HL,0EF00H      ;SET TOP MEM
000C 224940     00150         LD      (4049H),HL     ;  TO EF00H
000F C32D40     00160         JP      402DH          ;DOS
0012 02         00170 MACBUF  DEFB    02H            ;VIDEO ON
0013 56         00180         DEFM    'VERIFY (ON)'  ;VERIFY WRITES
001E 0D         00190         DEFB    0DH            ;"ENTER"
001F 43         00200         DEFM    'CLOCK (ON)'
0029 0D         00210         DEFB    0DH
002A 42         00220         DEFM    'BASICR'       ;LOAD DISK BASIC
0030 0D         00230         DEFB    0DH
0031 0D         00240         DEFB    0DH            ;# OF FILES
0032 36         00250         DEFM    '61000'        ;MEMORY SIZE. OPTIONAL
0037 0D         00260         DEFB    0DH
0038 52         00270         DEFM    'RUN"WATCH"'   ;HOME WATCHER
0042 03         00280         DEFB    03H            ;END OF STRING
0000            00290         END
00002 TOTAL ERRORS
```

Program Listing 4. Automatic Power-up to Disk Basic.

NOBODY KNOWS MORE ABOUT ADDING MEGABYTES TO YOUR TRS-80 MODEL II* THAN AMERICAN SMALL BUSINESS COMPUTERS

THE CAMEO CONNECTION

The Cameo 10 Million Character Disk Drive Subsystem (pictured in foreground) Is A 5 Megabyte Fixed, 5 Megabyte Removable Cartridge-Type Disk Drive. Advantages To This Unit Are Ease of Backup For Data and Program Files and Low Cost. By Attaching A Multiplexor Unit Several Model II's Can Be Attached To The Cameo Disk Drive. For Further Pricing, Availability, and Operating Systems Options, Call or Write to American Small Business Computers and ask for the CAMEO CONNECTION.



THE CORVUS CONNECTION

The Corvus Disk Drive Subsystem (center of picture) Is Available in 5, 10, and 20 Million Character Capacities. The Corvus System Is The Ultimate In The Sealed "Winchester, All-Fixed" Type Drive. The Corvus Is Designed To Operate In The Harshest Environments Such As Factories, Oil Rigs, Etc. Backup Of Data Is Accomplished Via Optional MIRROR Backup Device Whereby Data Is Written Onto A Standard Video Tape Recorder. A Great Number of Model II's May Be Attached To The Corvus Via An Optional CONSTELLATION Multiplexor. For Further Pricing, Availability, and Operating Systems, Call Or Write to American Small Business Computers and Ask for the CORVUS CONNECTION.

THE AMERICAN SMALL BUSINESS COMPUTERS MINI-WINCHESTER SPECIAL

The MINI-5 (being held by Corporate President Robert Webster) Is American Small Business Computer's Very Own Disk Subsystem. The MINI-5 Offers Approximately 5 Million Characters of Information Storage and is Based On the Famous Seagale Technology ST-506 Micro Winchester Drive. Now You Can Own State-Of-The-Art Technology At An Affordable Price — Only $3500 — Which Includes Drive, Controller Electronics, Power Supply, Cabinet, Cables, Documentation and Operating System Software.

DEALER INQUIRIES INVITED

AMERICAN SMALL BUSINESS COMPUTERS 529

*TM TANDY CORPORATION

118 SOUTH MILL ST. PRYOR, OK. 74361 918-825-4844

AMERICAN

118 SO. MILL ST • PRYOR, OK 74361
PHONE (918) 825-4844

ALL LISTED PRICES ARE CASH DISCOUNTED. WE ACCEPT MC, VISA, AND COD (REQUIRES CERTIFIED CHECK, CASHIER'S CHECK OR CASH). FOR CHARGE ORDERS, PLEASE ADD 3%. FOR COD SERVICE, PLEASE ADD 5%. SHIPPING AND HANDLING ARE FREE ON ALL ORDERS. OKLAHOMA RESIDENTS PLEASE ADD 4% SALES TAX.

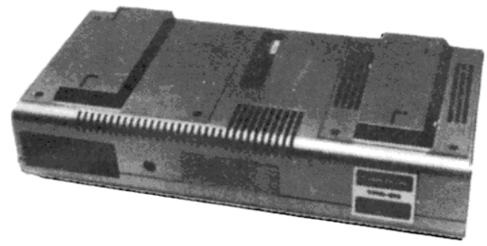
32K EXPANSION INTERFACES

SPECIAL OFFERING.

Due to a very special purchase, American Business Computers is able to offer a limited number of Radio Shack* Expansion Interfaces at the lowest price ever.

American Business Computers guarantees Expansion Interfaces to be Brand New — with original documentation and in perfect working condition.

35995



✓396

NEW PRODUCT FOR THE COLOR COMPUTER..."PACKET MAN" BY GREG ZUMWALT

In this exciting new adventure for the Color Computer*, you attempt to "eat" the dots and escape from the three little munchkins who are always chasing you. This game was written in machine code to take full advantage of the color computer's capabilities........ ....*Only $25.00*

MOON LANDER FOR THE COLOR COMPUTER *By Greg Zumwalt*..*Only* $14.95 ✓520

OTHER STUFF & ODDS — AND — ENDS

PAPER FOR YOUR PRINTER ✓521

9 1/2 x 11 — Tear-off perforations and you have an 8 1/2 x 11 "cut sheet". 3200 sheets of 15 pound paper....$39.95

CONTINUOUS ENVELOPES

Fan Fold Envelopes with tractor perforations down either side. Envelope size with perfs: 4 1/2 x 11"

PRICE........................59.95 per 1000

Price includes shipping in Continental U.S. ✓522

★ NOTICE ★

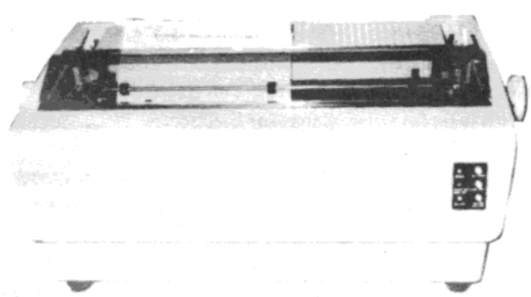
BY THE TIME YOU READ THIS, WE SHOULD HAVE OUR **DIRT CHEAP** MINI DISK DRIVES AVAILABLE FOR THE MOD III —

PRICE: Less than $599.95 ✓523

DISGUSTING!!

That's what we think about the price of most good "Letter-Quality" Printers. Our Starwriter is Bidirectional and out "specs" the others — hands down

$145000



✓524

ATTENTION:

CORVUS OWNERS

American Business Computers NOW offers a fully compatible TRSDOS* operating system. Our System, called CORDOS, runs all 10 MB, Revision B Corvus drives. The Program in this package contains a Master Program which will convert a user's operating system disk (Version 2.0) to an enhanced system capable of the following:

- Completely transparent use of the Corvus Hand Disk Drive
- Complete compatibility with existing programs
- All utilities (except Format and Backup, which should not be used) will operate normally.

Price: $300.00 with manual

— DEALER INQUIRIES INVITED —
CORDOS Author-Andy Frederickson
*TM of Tandy Corporation
*TM of Corvus Systems

✓525

See List of Advertisers on page 386

When your program crashes you can still save what's in memory.

Debug With GOTO

Roger L. Pape
7545 Marble Drive
Liverpool, NY 13088

You've just spent hours entering data into a program and then it "bombs out." Is there a way to continue without losing all your data? How about making changes in the source statements of the program and continuing without losing the data? Or would you like to run one program and have it load another program which uses values from the first?

Problem and Its Solution

Preserving a lot of data already in memory became a real problem for me one evening. After several hours entering a large number of long character strings, I was confronted with an OM Error message as the program terminated. Typing RUN would wipe all the data out! I observed that the point after the run command performs its initialization is equivalent to the GOTO statement processing. One can type GOTO nn (where nn is any existing line number) and reenter the program at the statement without losing the variables already stored in memory. A simple GOTO the cassette output section of the program and my data was saved. Reading the Level II manual later, under the description of the GOTO statement (section 4) was the solution I had discovered.

A GOTO in the command mode is a perfectly valid way of starting or reentering a program. Any variables created in the previous execution or those defined by assignment (Let) statements in the command mode before starting are preserved. The GOTO can be directed to any valid line number in the program with several restrictions. First, the Clear statement will clear the variable storage area (actually only resetting the storage pointers to the beginning of the storage area). If the Clear statement is executed when the program is re-entered, previously generated values are lost. Secondly, a variable cannot be redimensioned. Once an array has been created and stored, do not reenter at or before a Dimension statement containing that variable name.

Debugging with GOTO

GOTO can be used very effectively when debugging programs. Consider the following situation where a statement such as:

```
200 A = X/Y
```

is somewhere in the middle of a program and the value of Y is undefined (or zero for any other reason). The program will stop at this point with a /0 (division by zero) message. To correct this problem, one can enter an appropriate value for Y and continue by typing:

```
Y = ____
GOTO 200
```
(both in the command mode)

and check the rest of the program. Frequently this can eliminate the need to reenter a long series of input that would otherwise be required to run the program from scratch.

Level I vs. Level II

After a program is stopped, the values of the variables used during execution remain in memory. Likewise, any variables defined in the command mode will modify the values previously stored or will be added to the storage area if the variable had not been encountered earlier. The only reason the variables cannot be accessed when the run command is reissued in a Level II system is because the pointers which define the storage areas are reset.

The Level I system software does *not* reset the variable storage pointers when the Run command is processed. Therefore, preset the value of any variable before the run command is typed and the values will be used in the program. But the Level II software resets the variable storage pointers when the run command is processed. So any predefined variables will be lost when a Run is issued. The preset variables in a Level II system, type the variable assignments statement in the command mode (as in Level I), but then type GOTO nn (where nn is a statement number at the beginning of the program) in order to start execution. Avoid any Clear statement in the program. If added string space is required, type the appropriate Clear statement before presetting the variables. (The amount of string space reserved by a Clear statement is remembered from one program to the next. It only changes when another Clear statement is executed or the memory size is set. After running a program which uses large amounts of string space, it's a good idea to type Clear to free some of the memory.)

Unfortunately, continuing execution is

"Once you are able to chain, you are free to edit a program and restart it without losing variables."

not as simple if the source program is changed. Whenever another program is loaded or when statements are inserted, deleted, or edited in a program that is already in memory, the variable storage pointers are reset. You could include a cassette load statement at the end of one program to load a second program, but variables from the first program are not accessible in the second program (unless written to some storage medium). Likewise, if any changes are made to the program statements, you cannot normally continue execution at some intermediate point (as in the division by zero example).

With a few software changes you can preserve variables in memory for use later. Since the procedures required to save the variables while editing the source statements are essentially the same as those needed to transfer values from one program to another when chaining, it is relatively efficient to provide both capabilities in the same pitch. Let's first review the memory allocation in a Level II system.

Memory Map

The way variables are stored in a Level II TRS-80 is illustrated in Fig. 1. This memory map shows the relative positions of the various storage areas in the user's memory as a program is loaded and executed. The values in parentheses on the left of the map are the locations (in hexadecimal) of the word pairs used to store the 16-bit addresses of the current starting point for each of the specified areas. The first location contains the lower byte of the pointer, while the next location contains the upper byte.

Variables are stored immediately after the Basic source statements. This area is subdivided into two sections. The first contains simple non-dimensional variables, while the second contains all the arrays (dimensioned variables). The pointer stored at locations 40F9/AH is the start address for the simple variables. The pointer stored at locations 40FB/CH is the start address for the array variables. Finally, the pointer stored at 40FD/EH is the start of the free space. The first of these three pointers is set after the program source statements have been entered and remain the same while the program is running. But the latter two pointers will change value during program execution as new variable names are encountered and their values are stored.

The Basic interpreter stores the variables sequentially in the order that they are encountered as a program is run. What gets stored is a combination of the variable type, name and value.

Each simple variable consists of a variable type flag (which is equivalent to the number of bytes needed to store the value), two ASCII characters representing the variable name (the second character of the name, if it is used, or else a null is stored first), and the current value of the variable.

Array storage also includes overall length and dimension information. The name is followed by a two-byte value representing the remaining bytes for the array, then one byte for the number of dimensions and two bytes for each dimension to store its size (the last dimension is stored first). As a variable name in a program statement is interpreted, the storage area is scanned from the beginning to see whether this name and corresponding type are already stored in the list. If not, the variable is added at the end of the appropriate area. Array storage space is created when a dimension statement is interpreted. That is, the variable name is entered in the list and the necessary space reserved with the contents initialized to zero. If a new array name is encountered in any other statement (without previously being dimensioned), it is automatically dimensioned with an upper limit of 10.

Because simple variables and arrays are stored in separate areas, one can use an identical name for both. In addition, if the explicit type flags are used with the name, separate variables are created since both the name and type must match. That is, X%, X#, X!, and X$ are all distinct variables. Therefore, it is possible to have a number of different variables with the same one or two-character name. The Basic interpreter keeps them straight because it can determine whether the variable is dimensioned from the use of subscripts and also determine the variable type, based on the default type corresponding to the first letter or from an explicit type flag. All these conditions must match an entry in memory or else it is assumed to be a new variable and a new entry is created.

Scanning the variable list can involve considerable processing time. Variables are not stored in any prescribed order. Integers, single and double-precision floating point values, and string variable parameters are intermixed. So the search for a variable involves beginning at the start of the storage area and examining each variable name and type until a match is obtained or the end of the area is reached. In the case of simple variables, the variable type flag conveniently provides the number of bytes to skip over for the start of the next variable.

In the case of arrays, its length, stored

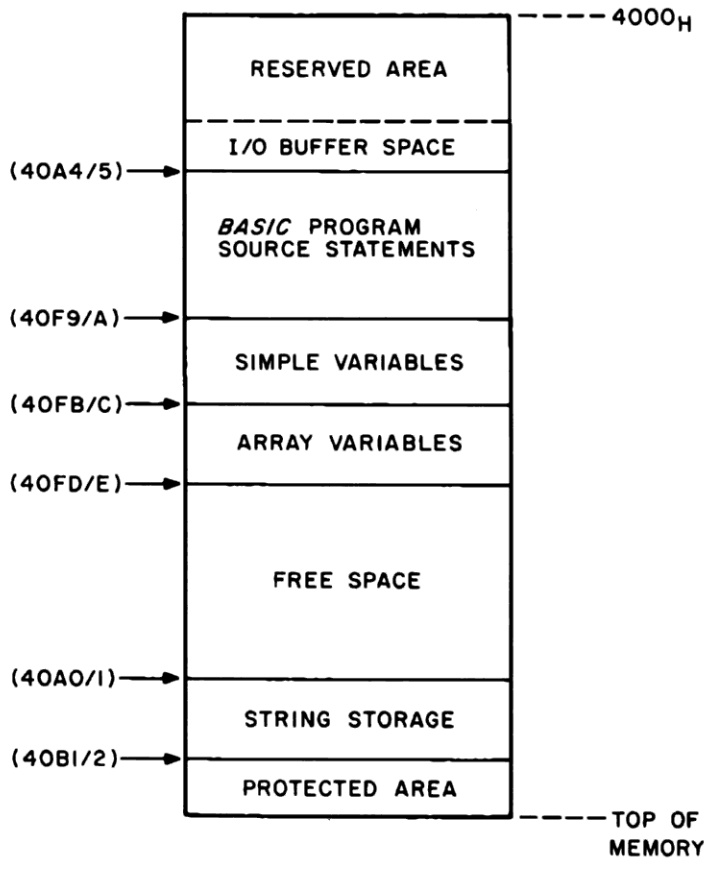


Fig. 1

"With a few software changes you can preserve variables in memory for use later."

immediately after the name, functions the same way. To save time during the scanning process, this value is computed once when the array storage space is first created. The number of bytes that follow is given by the byte length for the variable type times the product of the dimension, plus the number of bytes needed to store the dimension information. Then, when arrays are scanned and the variable name or type does not match, this value is read and quickly locates the start of the next array. Since the storage area must be scanned each time a variable is used, a time-saving tip is to declare the most frequently used variables early in the program so they are stored near the beginning of the list. Potentially, this could save considerable execution time in lengthy programs with large amounts of repetitive calculations.

Because the variable storage area immediately follows the program statements, increasing the program length, either by editing or loading a new program, will overwrite the old variable storage area. Any time a program is loaded or an existing program is edited, as well as when the Run command is issued, the pointers which define the extent of the variable storage sections are all initialized to the location after the end of the program code. This is interpreted as an empty storage space.

One other point that will effect your program is the way string variables are stored. The string information stored in a variable storage block consists of the type flag (3), the number of characters in each string and the start address for the actual string. This information is intermixed with the other variable types in one of the two variable storage areas, depending on whether it is a single string or an array of strings. If a string variable is entered via an input statement or is generated by a string function or by the concatenation of several strings, the actual string of ASCII characters is stored in an area of upper memory reserved for strings. But if the complete string is defined in an assignment statement of the program, such as:

```
500 A$ = "THIS IS A STRING"
```

or in a data statement, the string is left in the program area. This approach conserves storage space, but results in strings being scattered throughout memory. If strings stored in one program are to be used by another chained program, it is necessary to extract strings from the program area and save them in the string storage space.

The Chain Command

So, these are the required steps for chaining in Level II:

- Store all variables, including embedded strings, in the upper part of memory and protect them.
- Load the new program.
- Move the saved variables back down immediately following the program statements.
- Start the new program *without resetting the variable storage pointers.*

The first three steps in this process are illustrated in Fig. 2. Relocating the strings that are embedded in the program statements can be handled quite easily with an existing system routine in ROM. As these strings are moved, it may be necessary to expand the string storage area, so the string relocation should be performed before the variable block is moved.

Once you are able to chain, you are free to edit a program and restart it without losing variables. Before making any changes to the program statements, perform step 1 above. The editing or statement additions/deletions would be made (in place of step 2) using the normal system commands. After all changes have been entered, steps 3 and 4 are performed to repack the variables at the end of the corrected program and restart the program at the desired point.

Summary

Chaining is useful in a variety of situations. For example, you may have a program requiring the results of a previously run program. Rather than modify an existing program, it may be more convenient to write a separate program called by a Chain command to preserve the results of the first program. Breaking a large program into several smaller routines (which are chained) is almost equivalent to having more memory.

Another memory saving application involves programs with extensive internal data storage. Once data statements are read, they are essentially wasted space. An alternate approach is to have a separate preliminary program consisting of read and data statements to preset the variables and then chain the main program.

The proposed chaining software also lets you resume your program after editing with little memory loss.

The concluding portion of this article will describe the actual Chain command for Level II, including the complete assembly code. It is completely relocatable and automatically installs itself immediately below any other patches previously loaded, such as key debounce, lowercase, or whatever other fixes you may already have in your system.■

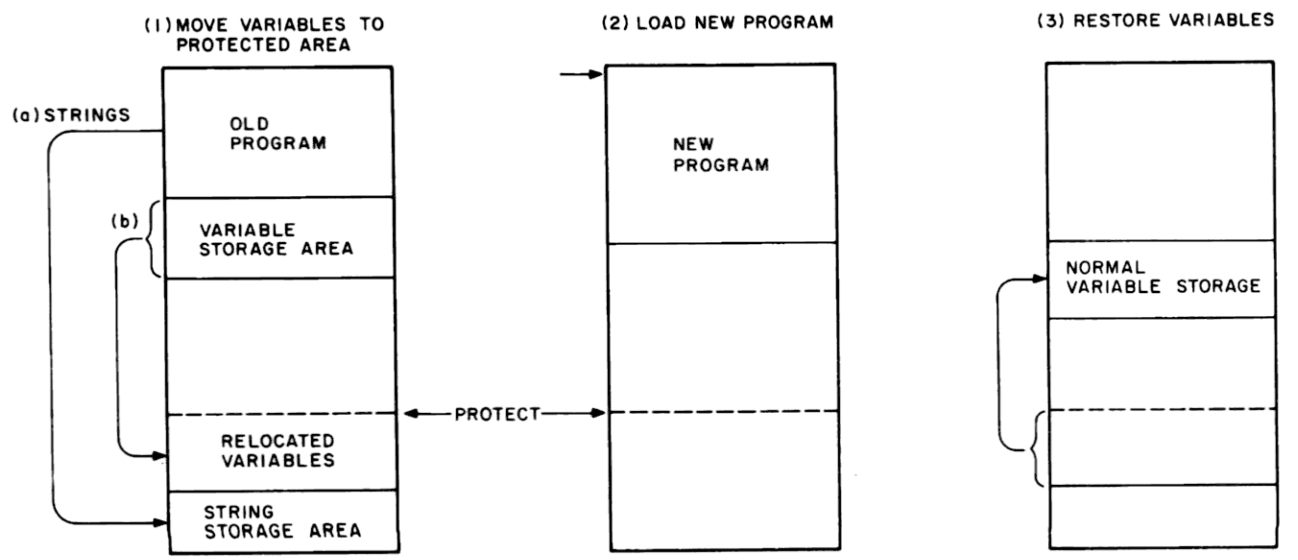


Fig. 2

At home on the Mod I and III.

DOSPLUS 3.3

DOS PLUS 3.3
Microsystems Software, Inc.
Hollywood, FL
$99.95

James LaSalle
M.R. 2 Bock Lane
Baden, PA 15005

At this writing there is only one serious double-density DOS available: DOSPLUS 3.3. It not only reads single density TRSDOS, LDOS and NEWDOS/80 diskettes, but reads standard Model III TRSDOS diskettes; *and* the Model III DOSPLUS reads and writes diskettes that are compatible with the Model I.

However, specialized disk I/O files and hardware-dependent features cannot be shuttled between the machines. The little gem that lets the Model I read Model III TRSDOS diskettes is the Convert/CMD utility. Convert :1 :0 makes the required adjustments to allow access to Model III diskettes.

DOSPLUS 3.3 allows mixing of various track drives. The Format/CMD utility formats disks for the desired track count and storage mode (double or single density). Once this is done, DOSPLUS 3.3 takes care of all housekeeping. LDOS/VTOS handles the different track drive problem in the same manner. If specifying the track on each format is disturbing to you, Config will allow you to preselect the track count and stepping rate, operate with a clock speed modification installed, configure a drive as double sided, (you can only configure the system drive to a certain track count) and if your printer can handle them, send unaltered graphics codes to the printer.

Transfer Your Library

You may want to transfer your program library from single to double-density format. Unlike Percom's DBLDOS that requires you to manually copy all filespecs, DOSPLUS does this with the Transfer/CMD utility. DOSPLUS reacts to the disk format (double or single density) and transfers files from disk to disk in the appropriate mode.

First format the destination disk in double density and execute Transfer :s :d (:s equals source drive, :d equals destination drive). DOSPLUS moves all visible files with no user intervention.

Directory

LIB gives you a look at a respectable library of DOS commands. Many of the familiar ones have been extended. DIR typifies the library enhancements. With most DOSes (LDOS excepted) the directory lists only the visible files on a disk; however, with DOSPLUS 3.3 the information is much more thorough.

The first line of the directory lists the drive number, disk name and date, version of DOSPLUS and density mode. The files are then described with various parameters. The ATTRB column first tells if the file is visible, invisible or deleted. The asterisks denote a user file. An S in the second ATTRB column would denote a system file. Next a U, A or B tells if the update, access or both passwords are set. The last ATTRB column tells the protection level set.

The directory also indicates the number of records on file, number of sectors, number of granules used, and if the file is written on consecutive tracks. The last directory line tells how much storage space remains on the disk.

A double-density disk filled with small utility programs easily fills several video screens in this format. This is not inconvenient because the system pauses after the video is full and waits for user instructions to proceed.

Other Commands

Free has been altered to graphically display the tracks currently being used on a specified disk instead of the amount of free storage left. This is a quick way to see how many tracks on a disk have been formatted where the directory is located and which tracks have been locked out.

Date can be set at power-up and is the default value for backups and format functions. DOSPLUS retains the date unless the system is rebooted. Copy has been streamlined to copy filename:s :d. This copies a file from drive :s to drive :d. Single-drive users are not forgotten. A special utility, Copy1/CMD, allows single-drive copying of all file types. Create preallocates disk space for a specified file. Debug can now be entered with a shift Break. Device displays current I/O devices and their vector addresses.

DOSPLUS 3.3 for TRS-80 Model I currently supports only three devices: keyboard, video monitor and printer. The RS232 is not defined at this time. Not implemented in this version of DOSPLUS are Link (I really miss this one), Set, Reset and the ability to route to a disk file.

Build-Do Commands

Chaining is available with the Build-Do commands. Build allows the user to create a sequential set of tasks for DOSPLUS to perform. A Do file may be created with Build to get a directory, display the Free map, load a printer driver, set Basic memory size, and jump to Basic. Build is completely self-prompting. A Pause command may be inserted in the Build file to allow for user intervention. Pause does not allow for user entry of commands.

Do checks high memory (4049H–404AH) and reserves about 300 bytes at the top of memory for itself and a small buffer. After

"There is only one serious double-density DOS available: DOSPLUS 3.3."

Do executes, this memory space is returned to the system.

A problem arises when a high memory program (printer driver, machine code sort, etc.) occupies the same memory area that Do usurps. Trying to Build a file that first loads the GSF (Racet computes) sort module into high memory is disastrous. When Do loads GSF, it obliterates its buffer and all Build instructions are lost.

The Forms command controls printer page length, the number of printed lines per page, the number of characters per line, and can output to a serial printer. It can also generate an automatic line feed on carriage return and perform a top-of-form.

A hard (printer generated) top-of-form is not implemented on my Okidata Microline 80 printer. Having to include paging modules in my programs has been a minor inconvenience in the past, but DOSPLUS 3.3 eliminates the need for extra page formatting code. On power-up the Forms parameters are set to standard values. To view the current status of the Forms driver type: Forms (enter). This also eliminates the need for any high memory printer routines. Since Forms is a library function these LPRINT/LLIST parameters are available for *all* printer output—even Basic listings!

The only minor irritant is that the TRS-80 Model I LPRINT line counter (found at 16425 decimal) is decremented with each printer code. In the Okidata Microline 80 printer expanded print is available by typing LPRINT CHR$(31); a return to normal print is LPRINT CHR$(30). The Model I and the Forms driver count each of these codes as a line printed and increment the LPRINT line counter. This could lead to a problem with printed output creeping down the page.

When I do not want Forms control I disable it by typing: Forms (L = 66, P = 66). This sets the page length and number of lines printed per page equal to 66. If I want a Basic program listing to accurately paged, I initialize the LPRINT counter by typing: LPRINT CHR$(12).

PROT has been expanded to allow the changing of a disk's name as well as the usual password manipulations.

Utility Functions

There are about a dozen programs listed in the utilities section of the DOSPLUS 3.3 manual—all of them extend the versatility of the system. By being discrete about which utility functions become programs, relatively small system disks can be created. This provides the single-disk user with maximum storage space per disk.

CLRFILE zeroes a file but does not reallocate its storage area. This could be useful when using a disk for temporary files. CLRFILE has the same overall effect as the library command Create.

Copy1 is a necessity for single-drive users. It permits single-drive disks to transfer all types of files. Both LDOS and NEWDOS/80 have this feature, but it is closely tied to the system. Copy1 will probably be little used by multiple-drive users.

Crunch is little changed from DOSPLUS 3.1. It is a utility to eliminate Remarks and extra spaces in Basic programs.

DiskZap has been amended to read single or double-density disks. Its capabilities are similar to Apparat's SuperZap. In its current form it does not generate hash codes or passwords. Conspicuous by its absence is SuperZap's DFS function. This permits examining or modifying by file name rather than by sector. Micro-Systems Software chose to provide this ability in a stand-alone machine language DiskDump program. Paging is slightly improved over SuperZap. Taken together these utilities should keep most disk zappers happy. Documentation is adequate for both utilities.

Purge allows you to rid a large number of files. Purge :0 (I) presents the user with a menu of visible and invisible files to be marked for deletion. Passwords are transparent to Purge so all files may be killed. With a handy program of this type it is easy to become over-zealous and kill the wrong file. The Restore utility, fortunately, resurrects the kill file as long as it has not been overwritten by another program or CLRFILE has not been used on it.

Map shows what tracks and sectors each file occupies. The value of this utility becomes evident when two disk systems of different track counts are used.

Spooler

Micro-Systems also provide an adequate spooler with DOSPLUS 3.3. The spooler uses an operator-defined memory buffer and an optional disk file for printer output. Spooling becomes a semi-background task allowing the host program disk I/O. This is accomplished by halting printer operation while the disk drives are being accessed. If the operator was too conservative in the allocation of spooling memory space, the spooler will temporarily seize CPU control until the spooler buffer is cleared of its backlog. Other than this, the operator is barely aware that the spooler is active. You

✓See List of Advertisers on page 386

"The SR/CMD module is unique. . .you expect to find this feature in a word processor."

can even list the spooler's disk file and printing still continues.

Deactivating the spooler requires rebooting the system. Since the spooler and the Do buffers can be co-resident, the spooler can be activated from a Build/Do file.

Disk Basic Version 1.4

DOSPLUS Extended Disk Basic Version 1.4 is authored solely by Micro-Systems Software, but I have not experienced a compatibility problem between DOSPLUS Basic and the MicroSoft/Tandy Extended Disk Basic. The only areas of incompatibility I encountered was with the specialized disk I/O file structure of NEWDOS/80 and with entry into Disk Basic itself. When entering DOSPLUS Basic, no file buffers (channels) are allocated. With other DOSes the default value is three buffers reserved. You must explicitly reserve I/O buffers when entering DOSPLUS Basic. The syntax for this also differs from the norm. The command: Basic filename-F:3-M:65237 loads Basic, reserves three I/O file buffers, protects memory above 65237 and loads and executes the program specified in filename.

DOS commands from Basic have been simplified to: CMD"DOS command". DOSPLUS 3.1 used CMD"I", "DOS command". A plain CMD returns to DOS. RENUM, SR and REF are called from Basic using CMD.

Basic has editing and keyboard shorthand. Pressing the semicolon displays the first line of code; pressing the slash key displays the last. Pressing L lists the entire program. The DU and DI commands allow the duplication and deletion/insertion of a program line. An L"filename" loads a program and a S"filename" saves it. Similarly R"filename" loads and executes a program.

Renumbering is adequate. To renumber a Basic program type: CMD"RENUM", n,i,s,e. Where n equals new line, i equals increment, s equals start line, e equals end line. Unlike NEWDOS/80, DOSPLUS 3.3 will *not* allow block moves via the RENUM command.

The SR/CMD module is unique in that it allows the programmer to search and replace a string variable or expression. You expect to find this feature in a word processor. I am sure this is one feature that future DOSes will incorporate.

REF/CMD is a variable cross reference. CMD"REF", K,L,V lists references in a Basic program by keyword, line number or variables. To obtain hard copy just append a P to the specification list.

CMD"M" is a kind of dynamic variables list. Stop a program at any time during execution and type CMD"M" and all variables are listed along with their current value. The debugging value of this command should be obvious.

The Trace function has been revamped to single step through Basic displaying program lines before executing them. This, along with CMD"M", has significantly reduced program debugging time for me. TRON turns on the trace and pressing any key single steps through Basic. Since only one instruction is displayed at a time, the video display is much easier to interpret. If Devices were fully implemented, the video monitor could be Linked to the printer to display the trace simultaneously on the monitor and printer. In the absence of Link the video could be Forced (Routed) to the printer.

Appending a ,V after a file to be run saves all current variable values. This allows passing variables from one Basic program to another. The result is true chaining. An index array could be established with an initialization program and passed to other programs.

File handling enhancements with DOSPLUS include variable record length files. This makes ISAM techniques possible. With variable record length files, the programmer no longer has to calculate subrecords. If a logical record length turns out to be only 100 bytes, then Open"R",1, "filename",100 opens buffer one for a record length of 100 bytes. Micro System's TESTPGM program amply illustrates this technique. Other than this sample program there is little explanation on the utilization of variable record length files.

In addition to the variable length files, Open"E" and Open"D" are available. Open"E" allows the extension of a sequential file without reading the entire file into memory and writing it back to disk. Open"E" causes new records to be added to the file on disk and the EOF marker adjusted. Open"D" seems to be synonomous with Open"R". It has been included to be compatible with TRS-80 Model II syntax.

Micro-Systems Software reworked Basic in DOSPLUS 3.3 quite extensively. Actually two Basics are provided: a full-feature Basic/CMD and an abbreviated TBasic/CMD. TBasic is meant to be used after a program is debugged and the various debugging utilities are not needed. This provides the user with more memory space for data. Not included in TBasic are the CMD function, editing and keyboard shorthand,

Daily Tribune

FORECAST: HOT

★★★★

1040 EDITION

RETURNS ARE IN; CPAIDS VOTED MVP

Title of "Most Valuable Program" Bestowed on Master Tax Package

Barely four years after its introduction into the microcomputer market, CPAids Master Tax software is being praised by hundreds of accountants as their Most Valuable Program.

Accountants nationwide have prepared thousands of returns using CPAids Federal 1040 software and today CPAids announced their enhanced programs for 1981.

The most notable addition to all CPAids software, the Fixed-Asset Accounting Program collects data year round and carries information over to the appropriate schedules and forms at tax time. Continuing an era of efficiency and cooperation, General Ledger II interfaces with the Master Tax Program to generate Schedule C. The Master Tax Program also generates a proforma tax organizer, providing necessary information from the previous years' returns.

The Time Is Right For CPAids

In an informal interview, CPAids dedicated its products to eliminating the frenzied feeling plaguing accountants during tax season and vowed to serve professionals well, not just at tax time, but all year.

Authorized dealers have pledged their continued support and offer a *free* 1981 update to those who subscribe now.

MASTER TAX

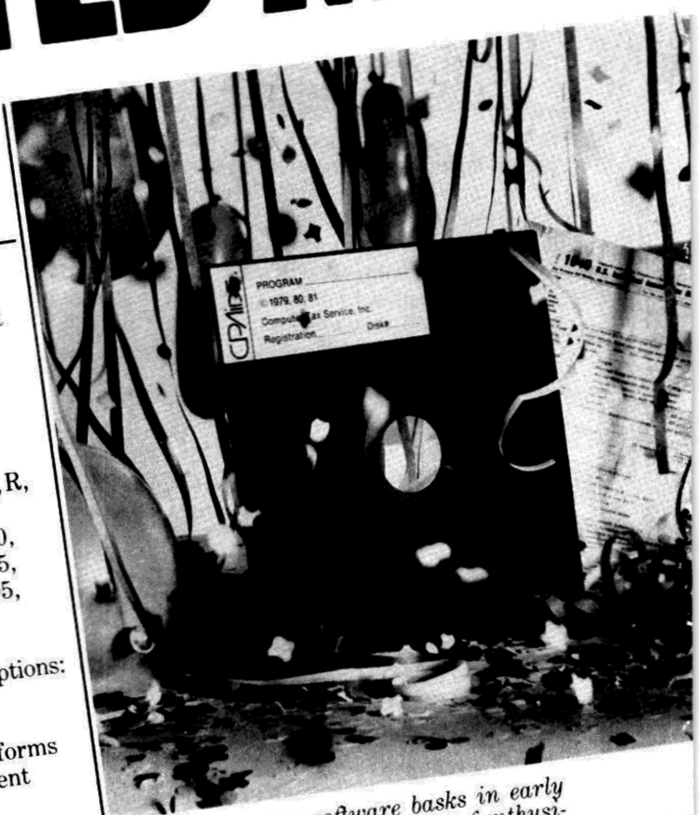
—Federal 1040 pages 1 and 2
—Schedules A, B, C, D, E, F, G, R, RP, SE, TC, ES
—Forms 1116, 2106, 2119, 2210, 3468, 3903, 4136, 4255, 4625, 4726, 4797, 4874, 4972, 5695, 5884, 6251, 6252
—Client instruction letter
—Exclusive CPAids print options:
- preprinted forms
- overlays
- generated computer forms
- printing on government forms

—Loan amortization program
—Input sheets

Also available this year, our new 1120 corporate program.

GENERAL LEDGER II

—General ledger, payroll fixed-asset accounting
—Loan amortization
—Stores all transactions
—Payroll interactive with the general ledger section
—Statement of changes in financial position
—Compilation letter
—Fixed-asset program stores up to 10,000 assets



CPAids 1040 tax software basks in early victory amidst cheering throng of enthusiastic users!

For only $55 CPAids will send you a demonstration disk and manual.

Call or write for free brochure and the name of the dealer nearest you.

CPAIDS®

1640 Franklin Avenue,
Kent, OH 44240 (216) 678-9015

Programs require a minimum memory of 56 Kb. Two disk drive, CP/M version 2.2 Runs in Microsoft Basic version 5.2 Master Tax program is available in compiled version only.

Computeri Improves as Tax Se

Studies show the intensity th complete a stu dimensions, th that IRS will r

No two syst it comes to pa three years t what Americ is on the way clients need the nation.

Despite t industry, it

"The shortcomings of DOSPLUS 3.3 are few and reflect my personal preferences. . . the literature is adequate for an experienced hacker."

or full error messages.

Shortcomings

The following shortcomings of DOSPLUS 3.3 are few and reflect my personal preferences: no DOS high memory command; lack of complete Device handling; no DOS Boot command for a warm reboot; the elimination of the Basic2 library command; and finally, only fair documentation. The literature is adequate for an experienced hacker, but may not suffice for an entry-level user. I especially miss a section on DOS calls.

There is also some conflict between DOSPLUS' keyboard routine and some Level II programs. Radio Shack's Haunted House adventure and Astrology are representative of this problem. Both programs were put on disk with Apparat's LMOFFSET. Executing either of these programs under NEWDOS/80 requires disabling the keyboard debounce routine. Small Systems Software's DCV-1 tape-to-disk utility solves this problem, however. DOSPLUS has no way to disable debounce so neither of these programs can be easily executed!

Assessments

My overall evaluation of DOSPLUS 3.3 is that it is a stable, efficient, fast and easy-to-master DOS. However, if you have mastered NEWDOS/80 you would probably miss many of the Apparat's extra enhancements (mini-DOS, block move of Basic code, etc.) if you adopted DOSPLUS as your bread-and-butter DOS.

The major factor in weaning me away from NEWDOS/80 was DOSPLUS' portability from Model I to Model III (and back again), and the ease with which it interchanges single and double-density diskettes. The standard features of DOSPLUS have been the basis for many stand-alone utilities. If nothing else was considered but this factor, DOSPLUS is a bargain. I have not used NEWDOS/80 with double-density zaps since acquiring DOSPLUS. A business can do little better than DOSPLUS for TRS-80 Model I or Model III.■

Author's Note: The Config command can only configure the system drive to a certain track count. Disks intended for drives with a track count different from the zero drive must be defined with the Format utility. The Config command may still be used to vary the stepping rate and number of disk sides for any drive as mentioned above.

Micro-Systems also tells me that Small Systems Software's DCV1 tape-to-disk utility will set the proper Level II keyboard values for correct operation. This means that DOSPLUS' debounce routine is no longer a serious problem for those few programs that require a Level II keyboard.■

Everybody's making money selling microcomputers. Somebody's going to make money servicing them.

New NRI Home Study Course Shows You How to Make Money Servicing, Repairing, and Programming Personal and Small Business Computers

Seems like every time you turn around, somebody comes along with a new computer for home or business use. And what's made it all possible is the amazing microprocessor, the tiny little chip that's a computer in itself.

Using this new technology, the industry is offering compact, affordable computers that handle things like payrolls, billing, inventory, and other jobs for businesses of every size…perform household functions including budgeting, environmental systems control, indexing recipes. And thousands of hobbyists are already owners, experimenting and developing their own programs.

Growing Demand for Computer Technicians

This is only one of the growth factors influencing the increasing opportunities for qualified computer technicians. The U.S. Department of Labor projects over a 100% increase in job openings for the decade through 1985. Most of them *new* jobs created by the expanding world of the computer.

Learn at Home in Your Spare Time

NRI can train you for this exciting, rewarding field. Train you at home to service not only microcomputers, but word processors and data terminals, too. Train you at your convenience, with clearly written "bite-size" lessons that you do evenings or weekends, without going to classes or quitting your present job.

Your training is built around the latest model of the world's most popular computer. It's the amazing TRS-80™ Model III, with capabilities and features to perform a host of personal and business functions. No other small computer has so much software available for it, no other is used and relied on by so many people. And it's yours to keep for personal or business use.

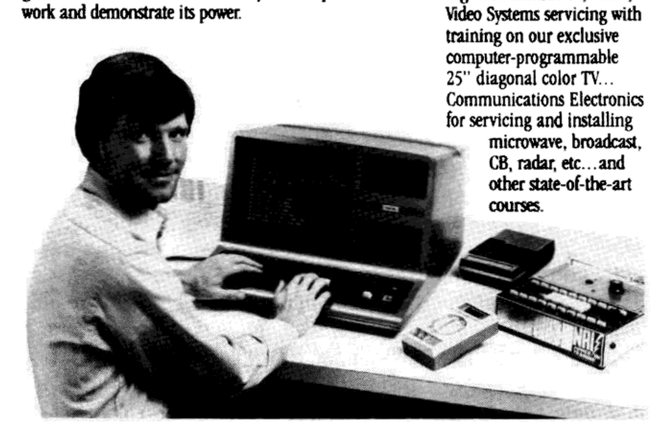
You get plenty of practical experience. Using the NRI Discovery Lab® that also comes as part of your course, you build and study circuits ranging from the simplest to the most advanced. You analyse and troubleshoot using the professional Beckman LCD digital multimeter you keep to use later in your work. Then you use the lab and meter to actually access the interior of your computer…build special circuits and write programs to control them. You "see" your computer at work and demonstrate its power.

Become the Complete Computer Person

You're also trained in writing and debugging both BASIC and advanced machine language programs…gain hands-on experience in the operation and application of computers to business and personal jobs. You're trained to become the fully rounded, new breed of technician who can interface with the operational, programming, and service facets of today's computers. You're ready to take your place in the new electronic age.

Other Opportunities

NRI has been giving ambitious people new electronic skills since 1914. Today's offerings also include TV/Audio/Video Systems servicing with training on our exclusive computer-programmable 25" diagonal color TV…Communications Electronics for servicing and installing microwave, broadcast, CB, radar, etc…and other state-of-the-art courses.



Rush coupon for Free Catalog… No Salesman Will Call

Send the coupon for our 100-page catalog showing all courses with equipment and complete lesson plans. There's no obligation other than to yourself. See how NRI can help you grow with the most exciting and important new field of the 80's. If coupon has been removed, please write to NRI Schools, 3939 Wisconsin Ave., Washington, D.C. 20016.

NRI McGraw-Hill

NRI Schools
McGraw-Hill Continuing Education Center
3939 Wisconsin Avenue
Washington, D.C. 20016

We'll give you tomorrow.

All career courses approved under GI bill.
☐ Check for details

NO SALESMAN WILL CALL.
Please check for one free catalog only.

- ☐ Computer Electronics including Microcomputers
- ☐ Color TV, Audio, and Video System Servicing
- ☐ Electronics Design Technology
- ☐ Digital Electronics
- ☐ Communications Electronics • FCC Licenses • Mobile CB • Aircraft • Marine
- ☐ Basic Electronics
- ☐ Small Engine Servicing
- ☐ Appliance Servicing
- ☐ Automotive Servicing
- ☐ Auto Air Conditioning
- ☐ Air Conditioning, Heating, Refrigeration, & Solar Technology
- ☐ Building Construction

Name ______ (Please Print) ______ Age ______

Street ______

City/State/Zip ______

Accredited by the Accrediting Commission of the National Home Study Council

179-101

Use them for more efficient handling.

All About Program Files

A. J. Barnard, Jr.
J. T. Baker Chemical Company
Phillipsburg, NJ 08865

A. David Barnard
Rensselaer Polytechnic Institute
Troy, NY 12181

Managing a small data base with cassette storage and a Model I can present problems. One problem is the low rate for data transfer and the limited reliability of the tape system. This problem is eased by hardware and software fixes such as the TC-8 cassette operating system.

A second problem stems from the nature of the records forming the data base. If the records are of variable length, alphanumeric, and not closely formatted, difficulties are magnified in the use of either data statements or a separately maintained cassette file; memory overhead for arrays and strings may be substantial.

With such records, a third approach can be considered—use of a program file.

The Program File

In a program file each record is stored as a statement against a line number. A program file becomes especially attractive if values for a significant parameter can be encoded as line numbers. For a program file, all the built-in, line-oriented subroutines of the programming language are available, notably the arrangement of records by the line-number-encoded parameter regardless of entry order. In this respect, a program file is a linked file.

A PEEK routine that steps through the file retrieves records from a Basic program file. When a line number in the search range is found, the associated record is recovered, manipulated and displayed by the string-handling capabilities of Basic.

Each record is entered against its appropriate line number just like a program statement with one important difference: A double quotation mark must be entered following the line number and before the proper record. The record is stored in the ASCII values of the characters and any embedded Basic key word (e.g., On, And, End) or control character is not interpreted as such. The double quotation mark ensures clear printing.

Retrieval of Records from a Basic Program File

The elements of a program to retrieve records from a Basic program file are highlighted by Program Listing 1. Line 0 clears space, principally for the string R, in which a recovered record is placed. Line 0 also defines the integer and string variables and then GOTOs the operating program at line 5000. Lines 1 through 4999 are thereby reserved for the program file. The first and last records, FR and LR, to be retrieved are input in response to lines 5000 and 5010. Line 5020 checks for entry errors.

The pointer to the start of a Basic program is at decimal address 16548 (40A4H) (least significant byte) and 16549 (40A5H) (most significant byte). For the standard Model I with 16K the location pointed to is 17129 (42E9H). This location X, and the next one, X + 1, hold the two-byte pointer to the address at which the next line begins.

The next two positions, X + 2 and X + 3, contain the two-byte value of the current line number, L. The statement associated with this line number begins at X + 4 and ends with a null byte (character code zero). This description explains the sequence of line 5030 (address, first line), 5040 (line number, current line), 5060 (if line number in scan range, reading of record into string R until null byte is reached), and 5090 (address, next line).

The string R is printed by line 5080 with the line number, and R is nulled for reuse. Line 5050 checks if the current line number is within the search range. If below, the scan continues by lines 5120, 5040, and back to 5050; if beyond, a new search is prompted (lines 5110, 5120 and back to 5000).

If the records of the program file, for example, held employee information by an alphabetically ordered time card number, the first two records might read:

```
1"AARON, MORRIS: HIRED 7/18/75; DEPT. 10
9"ALBERS, KARL: HIRED 12/15/60; ON LEAVE
```

A search for card numbers from one to nine or beyond would retrieve and display the same data.

This simple program can be applied to various small data bases by manipulating the string in which a retrieved record is placed. A practical restriction is that the program with an embedded file should CLOAD in not much more time than it takes to read an equivalent cassette file into a previously loaded program.

With 16K memory, an operating program should probably be less than 2.5K long and be packed by the use of multistatement lines, single-letter variables, DEF statements, etc., and by brevity in prompts and headings. The records should be telegraphically styled including abbreviations, acronyms and codes that can be replaced by full statements using If...Then...Else sequences or lookup arrays.

Program Listings 2 and 3 present two practical programs highlighting the use of Basic program files. They are tailored to the TRS-80, but can be adapted to other computers using Microsoft Basic. The listings have been made with line feeds and indents to improve legibility, but the programs should be entered without these niceties in order to conserve memory.

Daily Diary

Program Listing 2 is a daily reminder program. Lines 101 through 6031 are reserved for the program file and these numbers encode the month and day for a five-year period. For example, lines 101, 1301 and 4901 store the notes for January 1 for the first, second and fifth years. Year one is set as Y1 in line 10. The operating program consists of lines one through 85 and in execution requires only 1.8K of

"One problem is the low rate for data transfer and the limited reliability of the tape system."

memory with a record as long as 120 characters.

The search routine delineated by Listing 1 appears in lines 40, 45 and 50. The first and last dates (FD and LD) to be scanned are entered in response to line 20. By lines 20–25, the scan range is placed in string W, which is used as a heading. The dates are encoded as line numbers by the expression for C in line 25. Line 30 checks for entry errors and a response requires that the entered range is the intended one.

A retrieved record, as string R, is analyzed by lines 55 and 60. First, introductory spaces or double or single quotation marks are stripped. Then, if the third character in the resulting string is a colon, the first two characters are placed in string S, which is then checked by line 60 for a code. For example, if "HO: begins a record, then Holiday is substituted. The codes can be personalized. codes can be personalized.

Line 65 converts a line number to the month, day and year, which is displayed as string V ("##/##/##") (line 5) by the PrintUsing instruction.

Line 65, beginning with :MK = M, and line 70 form a day-of-week routine correct to year 2000. The abbreviation for the day taken from string T (line 5) is placed in string U by line 70. (To eliminate the day-of-week feature, delete the final statements of line 65, line 70, and the U; in the LPrint and Print statements of lines 75 and 80.)

If a printout is requested, flag H is set (line 35) and the LPRINT statements of lines of line 65, line 70, and the U; in the LPRINT and Print statements of lines 75 and 80.) and 75.)

The display is secured by line 40 (heading) and line 80 (records). When the counter J for the number of displayed records exceeds eight (line 80), press Enter for the next frame. After each record is printed, strings S and R are nulled for reuse. When the search is complete, line 85 prompts another. A daily diary file might start:

```
101"HO:NEW YR DAY-RUTH'S->DINNER
115"PD:IRS 4/4 EST TAX
117"CALL JIM ABOUT FEB SKI WKEND
```

If Y1 = 81 (that is, 1981) in line 10, then the CRT display for a scan from 1,1,81 through 1,17,81 would be:

```
-SCAN 1/1/81 ->1/17/81-
1/1/81-THR-HOLIDAY-NEW YR DAY-RUTH'S-DINNER
1/15/81-THR-PAYMENT DUE-IRS 4/4 EST TAX
1/17/81-SAT-CALL JIM ABOUT FEB SKI WKEND
```

If the diary becomes too long for a convenient CLOAD, the file for a completed period, say, six months, can be kept (with the operating program) as a separate tape by the use of the Delete and CSAVE instructions. Then only current entries need be updated and searched.

Stamp Inventory Program

Program Listing 3 is an inventory program for a philatelic collection. The collection of stamps, of course, must obviously be sufficiently valuable, extensive, or active to warrant computer management.

The operating program occupies lines 0 and 60000 on, and lines 1 through 59999 are reserved for the program file. The line numbers encode the catalog numbers of the postage stamps. The records need not

```
0 CLEAR 300: DEFSTR Q, R, S: DEFINT F, I, T
  : CLS: GOTO 5000
5000 INPUT "SCAN FROM RECORD "; FR
5010 INPUT"  TO RECORD "; TR
5020 IF FR < 1 OR TR > 4999 OR TR < FR THEN 5130
5030 X = PEEK(16548) + 256*PEEK(16549)
5040 L = PEEK(X + 2) + 256*PEEK(X + 3)
5050 IF L < FR THEN 5090  ELSE  IF  L > TR  THEN 5110
5060 FOR I = X + 4 TO X + 255
     : IF PEEK(I) > 0 THEN R = R + CHR$(PEEK(I)): NEXT
5070 IF LEFT$(R,1) = CHR$(34) THEN R =
     RIGHT$(R,LEN(R) - 1)
5080 PRINT L; "-"; R: R = ""
5090 X = PEEK(X) + 256*PEEK(X + 1)
5100 GOTO 5040
5110 INPUT "ANOTHER SCAN (Y/N)"; Q
5120 IF Q = "Y" THEN CLS: GOTO 5000 ELSE END
5130 CLS: PRINT, "ERROR OR MISMATCH"
     : PRINT: GOTO 5000
```

Program Listing 1

```
5 CLEAR300:DEFINTD-K,M,Y:DEFSTRP-W,Z
   :V="##/##/##":T="SATSUNMONTUEWEDTHRFRI"
   :P="DAILY DIARY -  BY A.J. & a.D. bARNARD"
10 Y1=81:'Y1 SETS YR ONE
15 CLS:PRINTP:PRINT,Y1+1900;"-";Y1+1904:PRINT
20 INPUT"SCAN FROM MM,DD,YY";D1,D2,D3:GOSUB25:SF=Z
  :FD=C:INPUT"  TO MM,DD,YY";D1,D2,D3:GOSUB25:SL=Z
  :LD=C:W="- SCAN"+SF+"-->"+SL+" -":GOTO30
25 Z=STR$(D1)+"/"+STR$(D2)+"/"+STR$(D3)
  :C=100*(D1+12*(D3-Y1))+D2:RETURN
30 IFFD<101ORLD>6031ORLD<FDTHENPRINT,"ERROR1"
  :GOTO20ELSEPRINTW;:INPUT" (Y/N)";Q
  :IFQ="N"THEN20ELSEIFQ<>"Y"THEN30
35 INPUT"PRINTOUT (Y/N)";Q:IFQ="N"THENH=0
  ELSEIFQ<>"Y"THEN35ELSEH=1:LPRINTW
40 CLS:PRINT,W:J=1:X=PEEK(16548)+256*PEEK(16549)
45 L=PEEK(X+2)+256*PEEK(X+3):IFL<FDTHEN50
   ELSEIFL>LDTHEN85ELSEFORI=X+4TOX+255:IFPEEK(I)>0
   THENR=R+CHR$(PEEK(I)):NEXTELSEGOTO55
50 X=PEEK(X)+256*PEEK(X+1):GOTO45
55 K=LEN(R):RL=LEFT$(R,1):IFRL=" "ORRL=CHR$(34)OR
  RL="'"THENR=RIGHT$(R,K-1):GOTO55ELSEIF
  MID$(R,3,1)=":"THENS=LEFT$(R,2):R=RIGHT$(R,K-3)
60 IFS="HO"THENS="HOLIDAY-"ELSEIFS="VA"THENS=
  "VACATION-"ELSEIFS="BI"THEN,S="BIRTHDAY-"ELSEIF
  S="PD"THENS="PAYMENT DUE-"
65 F=INT(L/100):G=INT(F/12.1):D=L-100*F:Y=Y1+G
  :M=F-12*G:MK=M:YK=Y+1900:IFMK<3THENMK=MK+12
  :YK=YK-1
70 N=YK+D+1+2*MK+INT((MK+1)*3/5)+INT(YK/4)
  :E=INT(N-7*INT(N/7)):U="-"+MID$(T,E*3+1,3)+"-"
75 IFH=1THENLPRINTUSINGV;M,D,Y;:LPRINTU;S;R
80 PRINTUSINGV;M,D,Y;:PRINTU;S;R:S="":R="":J=J+1
  :IFL=LDTHEN85ELSEIFJ<8THEN50ELSEPRINT@936,;
  :INPUT"HIT =ENTER= TO GO ON";Q:CLS:PRINT,W
  :J=1:GOTO50
85 INPUT"NEW SCAN (Y/N)";Q:IFQ="Y"THEN15ELSEEND
100 ' * LINES 101-6031: NOTES FOR 5 YRS *
```

Program Listing 2

"The collection must be sufficiently valuable to warrant computer management."

be formatted; with a colon separator however, fields can be displayed as separate lines and, beyond the first line, with an indent. In this way, different aspects of a collection are highlighted.

The Scott catalog system is the most common one in the United States. Unfortunately a Scott number consists of a numeral with one or more prefixed letters. The prefix denotes the stamp type (regular and commemorative issues have no prefix). This difficulty is resolved by the use of numeric offsets. For example, the first airmail stamp of any country has the Scott number C1; its record is stored against line 4401, that is, one plus the offset of 4400. The prefix/offset system established by lines 60020-60030 is adequate for the stamps of the United States and most countries of the British area of

```
0 CLS:GOTO60000
60000 PRINTTAB(8)"STAMP HOLDING"
      :PRINT,"PROGRAM - A.J. & A.D. BARNARD"
      :PRINT:CLEAR300:DEFINTF-K,M-N
      :DEFSTRP-W,Y:DIMT(44),M(44)
60010 S="(OWNER)-(MM/DD/YY)-(COUNTRY)->  "
      :P1="(#)":P2="(#)"
60020 FORI=0TO44:READT(I),M(I):NEXT
      :DATA"",0,AR,4000,B,4200,C,4400,CB,5400,
      CE,5500,CO,5600,E,5700,EO,5800,F,5900,

      FA,6000,J,6100,JQ,6500,K,6700,L,7000,
      LO,7300,M,7400,MB,7500,MC,7600,MO,7700,
      MR,7800,N,8000,NC,8400
60030 DATANE,8500,NJ,8600,"NO",8700,NR,8800,
      NRA,8900,O,9000,OY,9500,P,10000,PN,10300,
      PR,10400,Q,10600,QE,10700,RA,10800,
      RAC,10900,U,11000,UC,12000,UO,12100,UX,
      13000,UXC,13100,UY,13200,UZ,13300,"",13400
60040 PRINTS;P1;" - ";P2:M=0:PRINT
      :INPUT"SCAN SC # (WITH ANY PREFIX)";QF
      :INPUT"  TO # (WITH PREFIX)";QL:QT=QF:QP=""
      :GOSUB60050:QF=QP:N1=N:QT=QL:QP=""
      :GOSUB60050:QL=QP:N2=N:GOTO60060
60050 IFASC(LEFT$(QT,1))<=64THENN=VAL(QT)
      :RETURNELSEQP=QP+LEFT$(QT,1):QT=MID$(QT,2)
      :IFQT=""THEN60200ELSEGOTO60050
60060 IFQF="W"ORQL="W"THENQF="U":QL="U"
      ELSEIFQF<>QLORN2<N1ORN1<0THEN60200
60070 FORI=0TO44:IFQF=T(I)THENM=M(I)
       :MX=M(I+1)-M-1:GOTO60080ELSENEXT:CLS
       :PRINT,"NO PREFIX ";QF:PRINT:GOTO60040
60080 IFN2>MXTHENCLS:PRINT,N2;"BEYOND "
      ;QF;" RANGE":PRINT:GOTO60040ELSE
      W=QF+STR$(N1)+" - "+QL+STR$(N2)
      :N1=N1+M:N2=N2+M
60090 PRINT:INPUT"PRINTOUT (Y/N)";Q
      :IFQ="N"THENJ=0ELSEIFQ<>"Y"THEN60090
LSEJ=1:LPRINTS;W:LPRINT
60100 CLS:PRINTS;W:PRINT:F=0
       :X=PEEK(16548)+256*PEEK(16549)
60110 L=PEEK(X+2)+256*PEEK(X+3):IFL<N1THEN60180
       ELSEIFL>N2THENGOSUB60190:GOTO60040
       ELSEG=0:RF="":R="":FORI=X+4TOX+255
       :IFPEEK(I)>0THENR=R+CHR$(PEEK(I)):NEXT
60120 U=LEFT$(R,1):IFU=CHR$(32)ORU=CHR$(39)OR
      U=CHR$(34)THENR=MID$(R,2):GOTO60120
60130 FORI=1TOLEN(R):IFMID$(R,I,1)=":"
      THENG=G+1:RF=LEFT$(R,I-1):R=MID$(R,I+1)
      ELSENEXT:RF=R:R="":G=G+1
60140 IFG=1THENY=QF+STR$(L-M)
      ELSEIFG=2THENY=CHR$(LEN(Y)+192)
60150 F=F+1:IFF=9THENF=1:GOSUB60190
      :PRINTS;W:PRINT
60160 IFJ=1THENLPRINTY;" - ";RF
60170 PRINTY;" - ";RF:IFLEN(R)>0THEN60120
60180 X=PEEK(X)+256*PEEK(X+1):GOTO60110
60190 PRINT@935,;:INPUT"HIT =ENTER= TO GO ON"
      ;Q:CLS:RETURN
60200 CLS:PRINT,"MISMATCH OR ERROR"
      :PRINT:GOTO60040
```

Program Listing 3

"The Scott catalog system is the most common one in the United States."

specialty. If the number of offsets is altered, a change must be made in the upper limit of the For...Next loops in lines 60020 and 60070 and in the Dimension statement in 60020.

Since the program is tightly packed and without remarks, explanation is needed. The PEEK routine of Program Listing 1 is incorporated into lines 60100, 60110 and 60180.

By line 60000 space is cleared for strings (the amount can be decreased if no record exceeds 100 characters). Integer and string variables are defined to speed execution and conserve memory. String S in line 60010 must be personalized by owner's name, country, and date of file entry or update. This string serves as a heading for a printout (line 60090). The information is important for security or estate purposes. Scott numbers encompassed by the file are entered as P1 and P2.

Line 60040 displays the file heading and prompts entry of the range of stamps to be scanned, and with line 60050 separates each entry into its alphabetic prefix and numeral. (Line 60060 converts a prefix W to U; collectors of postal stationery will appreciate this feature.) A check for entry errors is then made.

```
4400 "LINES 4401-5399 AIRMAIL
4401 "U F LT CANC
4402 "U F:FFC(FULTON->NY12X25) ST
4403 "5 M F LH:4 U CDS
4406 "M F NH:6A SS M F:6B 11.5/11PERF U F FLT
4407 "9(NO 8A) M F LH:8A BID$85 STOLOW U F 15X180
4410 "M F LH:CVR FD CANC
4411 "M F NH TINY CR
4412 "U ON PC F 3 MGNS:U ON PCD G
4413  "15 M F LH:U F+:FDC SET ADDR F
4416 "M VG MISS PERF:ZEP CVR(FULTN->BERLN) F APPEAR
4417 "18 M G LH CR:TBECH PR M HH:17A BLU U VG
4419 "M VF:BLK6 M F:U F TINY TH
4420 "M F NH:PBLK4 M F NH:FDC PR UNADDR F:20A IMPERF U F
```

Program Listing 4

```
JOHN ADAMS-12/15/80-W. AMERICA-> SC C1-C20

1 - U F LT CANC
2 - U F
  - FFC(FULTON->NY12X25) ST
3 - 5 F LH
  - 4 U CDS
6 - M F NH
  - 4 U CDS
  - 6A SS M F
  - 6B 11.5/11PERF U F FLT
7 - 9(NO 8A) M F LH
  - 8A BID$85 STOLOW U F 15X180
10 - M F LH
   - CVR FD CANC
11 - M F NH TINY CR
12 - U ON PC F 3 MGNS
   - U ON PCD G
13 - 15 M F LH
   - U F+
   - FDC SET ADDR F
16 - M VG MISS PERF
   - ZEP CVR(FULTON-BERLN) F APPEAR
17 - 18 M G LH CR
   - TBECH PR M G HH
   - 17A BLU U VG
19 - M VF
   - BLK6 M F
   - U F TINY TH
20 - M F NH
   - PBLK4 M F NH
   - FDC PR UNADDR F
   - 20A IMPERF U F
```

Program Listing 5

By line 60070, the entered prefix for the scan range is compared with the array of offsets; the proper numerical offset is found and added to the numerals. Line 60080 checks if the upper limit of the range exceeds the assigned number of lines for a given stamp type; if not, the scan limits are placed in string W for a heading.

A retrieved record is placed by line 60110 in string R and analyzed by lines 60120 and 60130. Introductory spaces or single or double quotation marks are stripped. A check is then made for a colon in the resulting string; if one is found, the string to that point is placed in RF and the remainder left in R.

Flag G is set in line 60110. If a colon is found, G is incremented by line 60130. Line 60140 places the Scott number corresponding to the current line number into string Y (G = 1) for the initial display line for the record. For further lines, Y is a string of spaces equal to the number of characters in the Scott number.

The heading is displayed by line 60150 and the retrieved records by line 60170. When the counter F (lines 60100 and 60150) for the displayed record lines exceeds 10, Enter must be pressed (line 60190) for the next frame.

Line 60170 checks if the entire record has been printed; if not, the program returns to line 60120 for further analysis and printing of additional fields.

If a printout is requested (line 60090), flag J is set and the LPRINT statements of line 60090 (headings S and W) and 60150 (retrieved record) are executed. (If a line printer is not used, delete this feature.)

Use of this program is exemplified by Program Listings 4 and 5. Listing 4 is a contrived program file for airmail stamps of an imaginary country. Listing 5 presents a typical heading and the printout for this file. Stamp collectors will be familiar with the tricks used to compact records: well-accepted abbreviations, listing of sets versus single stamps, appending minor varieties that carry suffixed letters, etc.

The offset system can be extended to revenue stamps (e.g., with line numbers 20000 on). Secondary stamp-issuing entities can be included by added offsets; for example, the U.S. administration of the Philippines could be given the non-Scott prefix PHI.

This article has focused on the usefulness of Basic program files in the microcomputer management of small data bases with cassette storage. Program files effectively pack data and have special merit with loosely formatted records and where values for a significant parameter can be encoded as line numbers. ■

✓See List of Advertisers on page 386

Is anybody there... does anybody care?

Adventures in Modemland

Fred Blechman
7217 Bernadine Ave.
Canoga Park, CA 91307

In Part I of "Adventures in Modemland" I described how I got my TRS-80 connected to my telephone using the Microconnection modem from The Microperipheral Corp. Now I could communicate with microcomputer bulletin boards all around the country. Assuming I could only contact other TRS-80s I looked over a list of 133 bulletin boards from The Microperipheral Corp. and found several that looked TRS-80 oriented: Forum-80, Comm-80, Info-80, Hobbiest-80 and Engineer-80. The closest was Fullerton, CA—the Comm-80 bulletin board of the Orange County TRS-80 Users Group (OCTUG).

I reduced my equipment to the bare bones for modem operation. The Microconnection was ribbon-cabled directly to the TRS-80 keyboard card edge. My parallel printer and Exatron stringy/floppy tape storage unit, normally connected to the keyboard, were disconnected. I didn't want any interaction to create problems—I was confused enough already. See Fig. 1 for the minimum configuration.

Modem Encounter of The First Kind

I plugged in the Microconnection wall transformer, and made sure the telephone was connected. I loaded the S80 software program (provided with the Microconnection) into the TRS-80 using the cassette recorder. When the screen asked for (H)alf or (F)ull Duplex command, I typed in F. The screen went blank except for a cursor in the upper left corner. I put the Microconnection modem in the Simplex/Data mode, and random characters (created by the dial-tone) and line noise appeared on the screen. When I pressed a key that character appeared on the screen; this meant my modem was ready. I put the modem switches in the Duplex/Voice position, picked up the phone handset and dialed the OCTUG Comm-80 phone number, (714-526-3687). It answered almost immediately with a click and a steady tone.

I quickly set the Microconnection to the Duplex/Data mode. This transmitted a carrier of the proper frequency to the modem at the other end of the line, letting the Comm-80 system know I was in contact. I hung up the telephone handset and contact continued through the modem. After a couple of Enters from my keyboard, the Comm-80 started asking questions.

When OCTUG Talks, I Listen!

I had no experience with bulletin boards and no instructions of any sort. I just muddled through the questions, giving the best answers I could think of. The first question (Do You Need Linefeeds, indicate yes or no"?) I answered no. Then the screen gave a basic description of the system. It asked my first name, last name, and location. It repeated my inputs on the screen and asked for confirmation. The screen showed I was being logged onto Comm-80, with the date, time and caller number. When asked if this was my first time on the system, I answered yes. I was then asked if I wanted more information on the system; I replied yes and was told that Comm-80 is running on a TRS-80 with 48K, two MTI/MPI disk drives, TRSDOS 2.2, a Radio Shack RS-232-C, Centronics Micro P1 printer, a Novation CAT modem, auto-answer hardware from Computer Control, and software from Faulk and Associates Software.

This was followed by an invitation from OCTUG to use the system any time 24 hours a day, seven days a week. A simple command menu was then printed:

E—Enter a Message
K—Kill a Message
H—Help, Reprint this List
R—Retrieve a Message
S—List summary of messages
Q—Quick Summary (MSG # & Subject)
W—Redisplay the Welcome Message
T—Terminate the Session

Confusion and Panic

Although all this was happening at only 300 baud—about 30 characters per second—many lines were short. This meant that one, two or three lines a second were being generated on the bottom of the screen, pushing old lines off the screen. Without my printer connected, I had to read (and try to understand) too quickly for comfort. So I entered a T from my keyboard to terminate the session. The screen showed "User Logged Off Comm-80," gave the date, time, and total time on system (in hours, minutes and seconds) and disconnected. I put the Microconnection in Duplex/Voice model to disconnect at my end.

I called Info-80 in Seattle, but got hung up in the system, so I disconnected. After a couple of Forum-80 calls I started to get the hang of it. I found I was able to read messages within various categories (miscellaneous, personal, commercial, system bulletins) and I could enter messages of my own.

Non-Standardization

My confusion was caused mostly by the non-standardization among different systems. Various control keys and commands are used. The TRS-80 does not have a Control key; instead you use a combination of a letter key with an Up-Arrow, but not always. Some use Shift and Key or Down-Arrow and Key. Some just use the letter alone for control during receiving, but as a command when responding to a question. Without experimentation or documentation on a particular system, you can waste a lot of time (usually at long-distance rates). Command letters can be particularly confusing. To end a session, you might need a T to terminate, a B for bye, or a G for goodbye. Don't use an E for exit or end, since E usually means enter (for leaving a message). Sometimes a B means bulletins instead of bye. And G,

This is the second article by Fred Blechman about modems; the first was The Microconnection 80 Microcomputing, *August 1981.*

"... computer communication over phone lines is by standardized character code (300 baud ASCII)..."

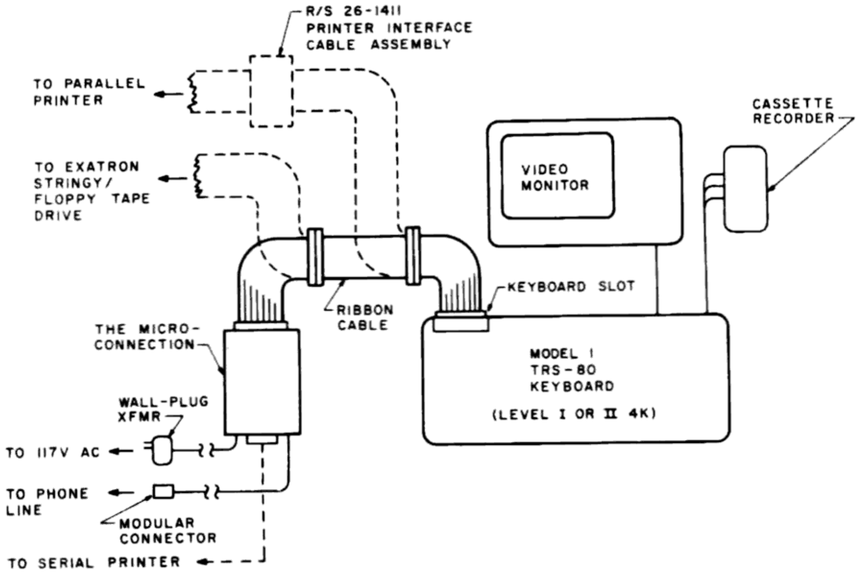


Fig. 1

Table 1
Typical Apple Bulletin Board System Commands

Function:
(A, B, D, E, G, H, K, L, N, Q, R, S, T, V, W, X, ?) ?H
Enter your Choice:
'ALL' For Complete Review
'CTRL' For CTRL Characters, or
'A, B, D, E, G, H, K, L, N, Q, R, S, T, W, X, ?' for Individual Response.
(C/R) to return to program
?ALL
Following is a brief list and description of the commands and their usage:

CTRL E—Retypes current line up to present position and allows you to continue from that point.

CTRL H (backspace)—Allows you to backspace one character at a time and prints A '↓' followed by the character you are backspacing over. This is the same routine as is used for delete or rubout instead of true delete. (For the benefit of printers)

CTRL U (forward arrow)—Starts you back at the beginning of the current line being typed (i.e. start over).

(C/R) to continue, (E) to end ?
A—Apple 40 column. Normally you would be allowed 64 characters per line. A bell will sound at 59 and on up to 64 columns at which point you would be forced on to the next line of text. In the Apple 40 mode, the bell will ring at 35, then again at 38 and 39, dropping you to the next line at 39. To avoid an extra blank line because of the 40th character, 39 was used instead of 40.

B—Print bulletin. Prints bulletins at beginning of program.

(C/R) to continue, (E) to end ?
D—Duplex Switch. Alternately selects full or half duplex operation and informs you of current status.

Table continues

used for goodbye on some systems, is used for graphics on Forum-80. Also, the same letter can have entirely different meanings as sub-commands within a single system.

Adding the Printer and Stringy/Floppy

With a few contacts under my belt, I decided to add the printer so I could have a printout of what was happening on the screen. I have an Okidata Microline-80 printer with a Centronics-compatible parallel interface, so I'm able to use a Radio Shack Printer Interface Cable (#26-1411) without modification.

Since I don't have an expansion interface and the Microconnection was already plugged onto the card edge at the left rear of the keyboard, I used a two-for-one bus extender from Exatron. This is a five-inch long 40-wire ribbon cable with a 40-pin male card-edge connector on one end, a 40-pin female card-edge connector on the other end, and a 40-pin male card-edge in the center. This allows you to connect two 40-pin cables to the keyboard in parallel. I connected both the printer interface cable and the Microconnection cable to the keyboard using a bus extender. To use the printer with the S80 software, press the Shift, Up-Arrow and P keys at the same time. To stop printing, press the Shift, Up-Arrow and S keys at the same time.

Next, using another two-in-one bus extender, I added the Exatron String/Floppy tape storage unit. I made a copy of S80 on a String/Floppy wafer, and it loads the S80 program in a few seconds. The system, with printer, Stringy/Floppy and Microconnection connected parallel to the keyboard, worked just fine. No crosstalk or interference has been noted even after several months of operation.

Talking to an Apple

I noticed several bulletin boards listed that were local phone calls—but they were all listed as ABBS. (Apple Bulletin Board System). I knew an Apple used a different Basic than a TRS-80, and that the screen on a standard Apple computer showed only 40 characters on a line, although the TRS-80 had 64 character lines. I dialed the Canoga Park ABBS at (213) 340-0135. Everything worked perfectly; I have since found that computer communication over the phone lines is by standardized character code (300-baud ASCII) using standardized frequency-shift keying (Bell 103), so any computers using these standards can talk to each other—Apple, PET, TRS-80, OSI, etc.

My first session with the Canoga Park system yielded a four-foot long printout in

Table continued

E—Enter message. Allows you to enter a message into system. Enter commands are basically self explanatory. A carriage return (C/R) at this point will list out the command menu for entries. The change command allows you to change an entire line but not just change part of it. Make sure when you are done with the message to save it to disc with the 'S' command.

(C/R) to continue, (E) to end ?
G—Goodbye. Exit program.

H—Help. Prints this routine.

K—Kill a message. Enter this to delete a message from the file. A password may be necessary if one was used at the time of message entry.

L—Line feed on/off. Normally on. For terminals that need an extra linefeed character to advance to the next line.

(C/R) to continue, (E) to end ?
N—Nulls. Adds an extra delay after a carriage return to allow printers time to move the printerhead back to starting position. This option only works with the line feed option on. Each null is equivalent to 30 milliseconds delay and is adjustable from 1 to 30. It defaults to one.

Q—Quick scan. An abbreviated scan See 'S'.

R—Retrieve messages. Allows you to retrieve a message from the file.

(C/R) to continue, (E) to end ?

Table continues

seven minutes and six seconds—and that included the time I spent scratching my head trying to figure out what to do next! Now I had the command and control codes for a typical ABBS system. (See Table 1 for main commands and control characters.) Additional sub-commands are described during specific functions if you hit Enter instead of entering a command.

I called another local system—the Woodland Hills ABBS at (213) 346-1849. This system included an automatic log-on function and a Chat mode. Typing a C as a command alerted the system owners (if they were available) who could use their Apple keyboard to send a message in real time; I'd answer from my keyboard. It's like RTTY (radio teletype) but uses the video screens instead of printers.

A Visit to the Big Apple

Since this system was located only a few miles away, I made an appointment to see it in operation. I couldn't believe how simple it was: An Apple II with 48K RAM, two disks, a 12-inch video monitor and a D.C. Hayes auto-answer modem are connected to a standard telephone. When it rings, the auto-answer modem triggers the computer. The disk-based program was purchased for about $40 from Rainbow Computing in August 1980. No user's guide is necessary, since the program is self-doc-

"Prepare to communicate with giants like The Source and CompuServe."

Table continued

S—Summarize messages. Allows you to scan over messages starting at the message # you specify.

T—Time and date. Gives you the current time and date. This is also used automatically during log-in.

W—Welcome. Prints welcome message at beginning of program.

X—Expert user. Does away with certain explanatory messages during the program. It also allows certain C/R defaults. Ex: A C/R in response to functions? will print functions supported by the system.

—Prints functions supported in that current mode of operation.

is self-documented and prompts the user.

Forum-80

While there are many bulletin board systems you can use with a TRS-80, the most widely available system designed with the TRS-80 user in mind—and using a TRS-80 as the host computer—is the Forum-80. Bill Abney, the father of Forum-80, spent two years writing the combination Basic and machine language programs. If you have an interest in setting up a bulletin board system, the current 3.1 version is available for a one-time license fee of $150, which includes free technical advice.

A minimum Forum-80 bulletin board system requires a TRS-80 with 48K RAM, three 35-track disk drives, an RS-232 board and an auto-answer modem. You can call Abney from 5 p.m. to 10 p.m. EST weekdays, or 10 a.m. to 7 p.m. weekends (816) 921-9439 for further information.

You can use a Forum-80 system by following the screen prompts and using the Help command. However, the system offers sophisticated features and many commands and sub-commands that can save you a lot of time and make your contact more efficient. To encourage proper utilization of Forum-80 systems, Abney distributes free user's guides, in two volumes. Volume I is the basic system and Volume II covers the more advanced features. For copies send a self-addressed, stamped, legal size envelope for each volume.

Postage is 28 cents for Volume I and 41 cents for Volume II. Mail to user's guide, Forum-80 Headquarters, 7600 East 48th Terrace, Kansas City, MO 64129.

Bulletin Board Roster

There are at least 200 bulletin board systems on line around the United States. Some lists have been published previously (*80 Microcomputing*, May 1980, page 110, and *Kilobaud Microcomputing*, October 1980, page 158) but the picture keeps changing. The most extensive list I know of is offered free; send a self-addressed stamped envelope to The Microperipheral Corporation, P.O. Box 529, Mercer Island, WA 98040—(206) 454-3303.

What Good Are The BBs?

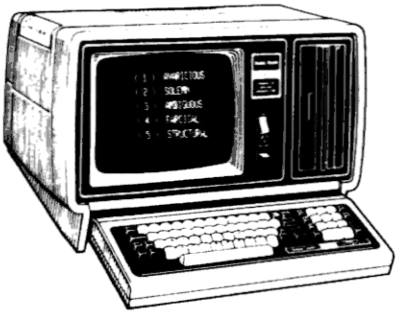
You might wonder why you should get excited about using a computer bulletin board when a phone call or letter might be more efficient. Consider this: You and your computer are in the forefront of a new technological explosion in the information and communications fields. Bulletin boards are a training ground for developing new communication techniques. Also, some allow programs to be uploaded (from your computer memory to the host computer memory) or down-loaded (from the host computer to your computer memory) in various languages. As this sophistication proceeds, so will your ability to use your equipment and knowledge. Furthermore, bulletin boards prepare you to communicate with giant information systems like The Source and CompuServe. ■

✓See List of Advertisers on page 386

MORE POWER TO YOU

We stock the complete line of hardware, software, and accessories

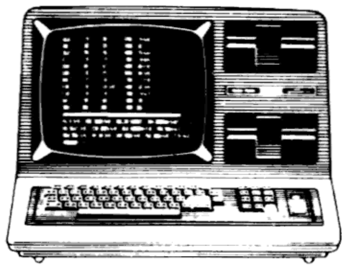
MODEL II



26-4002 64K 1 Drive
$3500.00

26-4160	1 Drive Exp	$1030.00
26-4161	2 Drive Exp	1560.00
26-4162	3 Drive Exp	2100.00

MODEL III



26-1061	4K I	$620.00
26-1062	16K III	875.00
26-1066	48K III W/2 Drives, RS232	2180.00

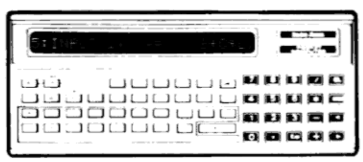
COLOR Computer or Video

26-3001	4K	$330.00
26-3002	16K Ext. Basic	480.00
26-3008	Joysticks	25.00
26-3010	Color Video	370.00
26-1206	Recorder	59.00

PRINTERS

26-1155	Quick Printer	$180.00
26-1167	9½ Dot Matrix Printer	325.00
26-1166	Line Printer VI	1025.00
26-1158	Daisy Wheel II	1725.00
26-1165	Line Printer V	1625.00

POCKET COMPUTER



26-3501	1.9K P.C.	$185.00
26-3503	Cassette 1F	49.00
14-812	Recorder	79.00

PERIPHERALS

26-1140	Expansion Interface	$250.00
* 26-1141	16K Expansion Interface with A.M. Ram	310.00
* 26-1142	32K Expansion Interface with A.M. Ram	350.00
26-1145	RS232C Board	84.00
26-1160/1	Mini Disk Drive	420.00

* Includes A.M. Electronics 90 Day Warranty

DEALER P124

R.S. 90 Day Limited Warranty
F-48 Form Provided

✓452

Fowlerville: 111 North Grand, Box 1071,
Fowlerville, MI 48836
(517)223-7281

Ann Arbor: 3366 Washtenaw Ave.,
Ann Arbor, MI 48104
(313) 973-2312

MasterCard

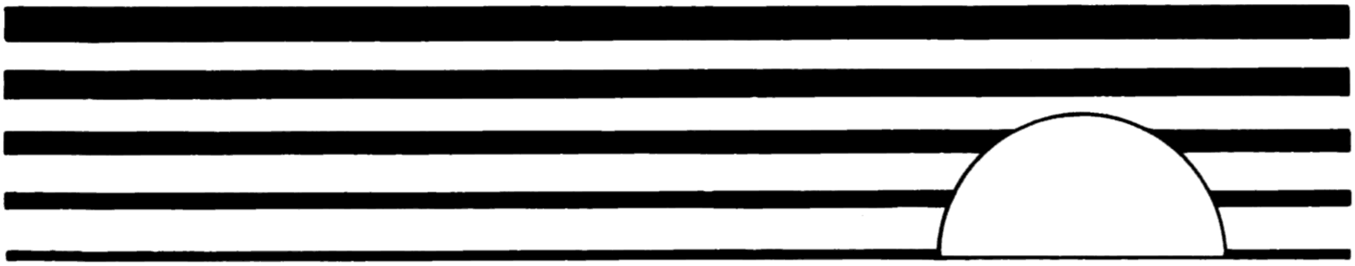
The power behind the drives©
A.M. ELECTRONICS, INC.

VISA

✓See List of Advertisers on page 386

Calculating time and place all year.

Sunrise . . . Sunset



Harold K. Skramstad
8045 S. A1A Hwy.
Melbourne Beach, FL 32951

Would you like to know the time of sunrise and sunset without having to look it up in the daily newspaper? The following program finds the declination of the sun, the equation of time (the difference between apparent and mean time), the compass direction of the sun at sunrise and sunset, and the time of sunrise and sunset for any point in the continental United States.

If you are interested in these values for only your own location, the program can be simplified by replacing the statements before line 210 with statements giving the values of LA—your latitude in degrees and tenths, and TD—the time difference between sun time and standard time for your location.

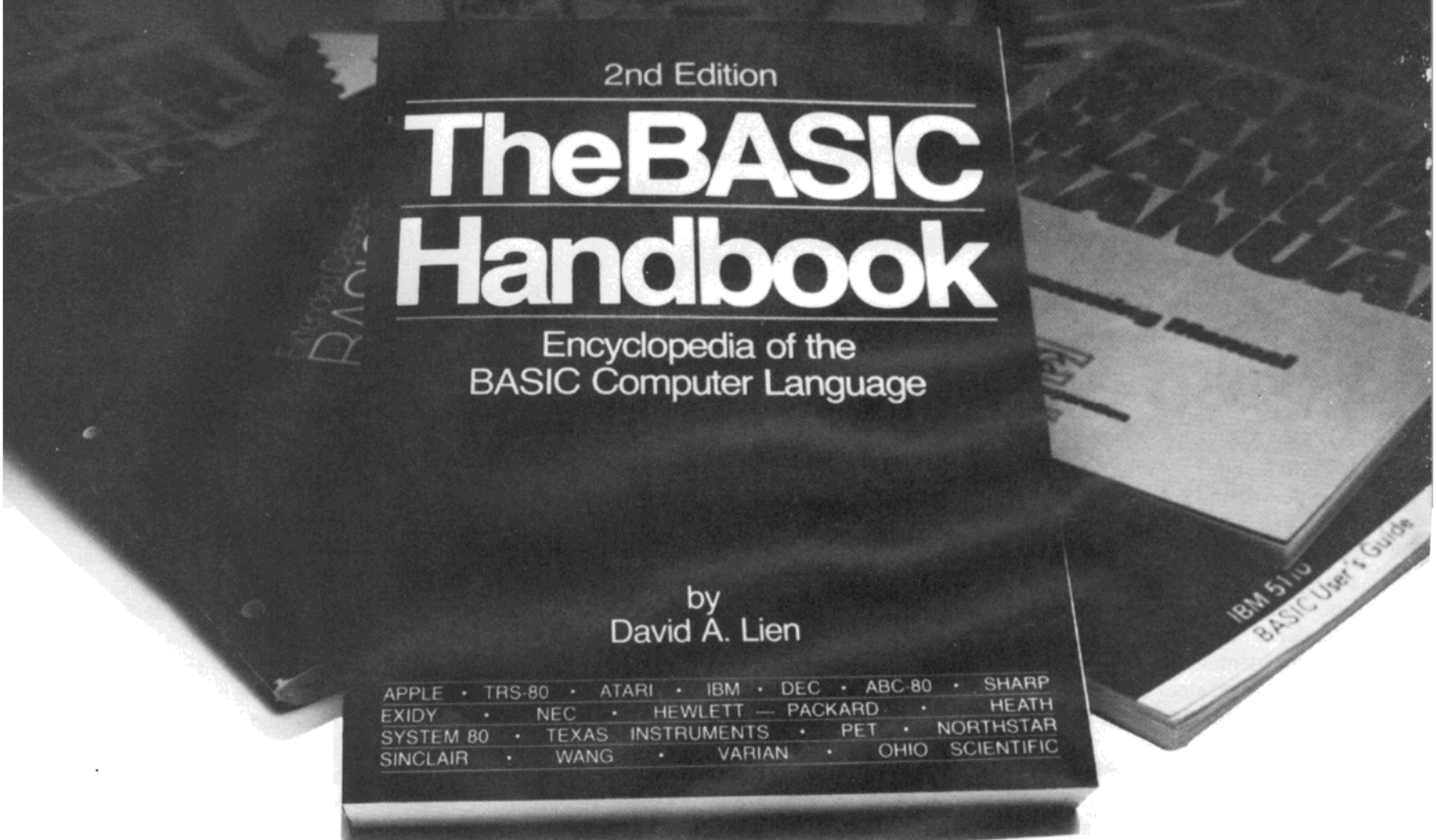
The declination D and the equation of time E are approximated by finite Fourier series in terms of X, the week number and fraction of the week, in lines 280 and 340. The value .04 in lines 530 and 560 is a correction for the refraction of light in the earth's atmosphere at sunrise and sunset.

I have found this program to agree within two minutes with the times of sunrise and sunset given in the local newspaper. ■

Program Listing

```
10 PRINT"THIS PROGRAM FINDS THE DECLINATION OF THE SUN, THE EQUA
TION"
20 PRINT"OF TIME, THE AZIMUTH ANGLES OF SUNRISE AND SUNSET, AND
THE"
30 PRINT"TIMES OF SUNRISE AND SUNSET FOR ANY POINT IN THE CONTIN
ENTAL"
40 PRINT"UNITED STATES."
100 DIM N(12)
110 PL=3.14159/26: J=57.29578
120 INPUT"ENTER LATITUDE (DEG.,MIN.)";D1,M1
130 LA=D1+M1/60
140 INPUT"ENTER LONGITUDE (DEG.,MIN.)";D2,M2
150 INPUT"ENTER TIME ZONE (E,C,M,P)";T$
160 IF T$="E" THEN LO=75: GOTO 210
170 IF T$="C" THEN LO=90: GOTO 210
180 IF T$="M" THEN LO=105: GOTO 210
190 IF T$="P" THEN LO=120: GOTO 210
200 GOTO 150
210 TD=(D2+M2/60-LO)/15
220 INPUT"ENTER MONTH NO., DAY OF MONTH";M,DA
230 FOR I=1 TO 12: READ N(I): NEXT I
240 DATA 0,31,59,90,120,151
250 DATA 181,212,243,273,304,334
260 X=(N(M)+DA)/7
280 D=.456-22.915*COS(PL*X)-.430*COS(2*PL*X)-.156*COS(3*PL*X)+3.
830*SIN(PL*X)+.060*SIN(2*PL*X)-.082*SIN(3*PL*X)
300 PRINT
310 PRINT"DECLINATION OF SUN:";
320 PRINT USING"###.#";D;
330 PRINT" DEGREES"
340 E=.008+.510*COS(PL*X)-3.197*COS(2*PL*X)-.106*COS(3*PL*X)-.15
0*COS(4*PL*X)-7.317*SIN(PL*X)-9.471*SIN(2*PL*X)-.391*SIN(3*PL*X)
-.242*SIN(4*PL*X)
360 PRINT"EQUATION OF TIME:";
370 PRINT USING"###.#";E;
380 PRINT" MINUTES"
390 CL=COS(LA/J): SD=SIN(D/J): CD=COS(D/J): Y=SD/CL
400 IF ABS(Y)=>1 THEN PRINT"NO SUNRISE OR SUNSET": END
410 Z=90-J*ATN(Y/SQR(1-Y*Y))
420 PRINT"AZIMUTH OF SUNRISE:";
430 PRINT USING"####.#";Z;
440 PRINT" DEGREES"
450 PRINT"AZIMUTH OF SUNSET: ";
460 PRINT USING"####.#";360-Z;
470 PRINT" DEGREES"
480 ST=SIN(Z/J)/CD
490 IF ABS(ST)>=1 THEN T=6: TT=6: GOTO 520
500 CT=SQR(1-ST*ST)
510 T=J/15*ATN(ST/CT): TT=T
520 IF D<0 THEN T=12-T: TT=T
530 T=T+TD-E/60-.04
540 GOSUB 600
550 PRINT"TIME OF SUNRISE:";T1$;":";T2$;" ";T$;".S.T."
560 T=12-TT: T=T+TD-E/60+.04
570 GOSUB 600
580 PRINT"TIME OF SUNSET: ";T1$;":";T2$;" ";T$;".S.T."
590 END
600 T1=INT(T): T2=T-T1: T1$=STR$(T1): T2=INT((T2*600+5)/10)
610 T2$=STR$(T2): T2$=RIGHT$(T2$,LEN(T2$)-1)
620 IF INT(T2)<10 THEN T2$="0"+T2$
630 RETURN
```

THE GREAT SIMPLIFIER!



The only book that simplifies the BASIC language by explaining its many dialects is now even better! We've added 240 BASIC words from the world's latest computers, for a new total of nearly 500 words detailed in a concise easy-to-read manner.

The 480-page **2nd Edition BASIC Handbook** tells you everything you need to know about today's BASIC language. It explains *all* significant Statements, Functions, Operators and Commands, so you can put them to work instantly!

If there's an alternate way to write a program using other BASIC words, the **Handbook** shows you how. If a program needs a Function your computer doesn't have, the **Handbook** generally has a subroutine which does the same thing. And it's all alphabetized and indexed for fast access!

The big *new* **2nd Edition** helps you convert those "strange" programs found in magazines and professional journals to RUN on your computer. Extra sections describe many specialty BASICs, including Atari, Sinclair, Tektronix, TRS/Color, plus Disk BASIC.

For students, businessmen, hobbyists, and pros — anyone who works with BASIC, the updated **Handbook** is indispensable. Order your **Second Edition** today! It's the missing link between you and more computer power!

Yes! Give Me More BASIC Power!

Please send ________ copies of **THE BASIC HANDBOOK 2nd Edition.** (30 day money back guarantee)

☐ My check for $19.95 each and $1.65 P&H is enclosed. (Californians add 6% tax.)

☐ Charge it to my MasterCard or Visa (circle one).
Acct. No. ____________________ Exp. Date ________

Name ______________________________

Address ______________________________

City ____________ State ________ Zip ________

Signature ______________________________

Mail to: **CompuSoft® Publishing,** *1050-E Pioneer Way, Dept. A*
El Cajon, CA 92020

Or Call: **(714) 588-0996** to order now! ✓377

Where you'll be with this meat inventory program.

Fat City

David D. Busch
515 E. Highland Ave.
Ravenna, OH 44266

Letting your computer keep track of the meat in your home freezer can save frostbitten fingers and energy lost to escaped cold air. Better yet, firm control over what you have and don't have can keep you from accidentally dipping into the latest meat stocks while an older cut languishes at the bottom of the stack.

Freezer Inventory was written to take some of the confusion out of stocking and maintaining meat in a home freezer. Some homeowners use log books which are difficult to maintain, never up-to-date, and time-consuming. It's usually necessary to do a physical inventory once each six months or so just to be sure of what's hidden in dark recesses of the freezer.

Others stock and replenish by the seat of their pants, knowing vaguely how much ground beef remains, and frequently allowing choice cuts to succumb to freezer burn or some other malady because they were kept two or three years beyond the recommended freezer life.

Features

Freezer Inventory is a disk-based meat inventory system written for the TRS-80 Model I or III. Type, cut, weight and date frozen for up to 50 pieces of meat can be entered and stored. Depending on memory available, the number of items in the inventory can be increased.

In accessing the data, the user can see a list of all the foods in the freezer or scan the available cuts of only one type of meat. Nervous types can also retrieve inventory items by age. The program presents a list of all meats that were frozen one to six months prior to the current date.

Meats can be entered in one of seven different categories: pork, beef, lamb, fowl, ham, fish and other. Those who normally do not eat a certain type of meat for religious or other reasons can delete that selection(s) from the list. On the other hand, if rattlesnake, game or goat is part of your menu, it can be added.

Thirteen different cuts can be entered to further differentiate the item in the meat inventory. I put together the list to conform to the cuts we most frequently buy in our household: three kinds of roasts, three kinds of steak, chops, and other configurations such as ground. Again, these may be altered by the user.

To initialize the program, the data file must be created. This can most easily be done by typing DIM FOOD$(50,4) in the command mode, and then typing GOTO 1420. The program will open the file Foods, and save the available data (nothing, at this point) to the disk. Then, a File Not Found error will be avoided when the program is first run.

Data on the different meat types and cuts are Read into two string arrays, Type$(n) and Cut$(n), at the beginning of the program. Next, the user is confronted with a menu, which presents six options:

- See list of all foods in the freezer.
- See list of only one type of food in the freezer.
- Add new foods to freezer inventory.
- See list of foods by age.
- Save information input this session to disk.
- Remove an item from inventory.

Entering New Food Data

Depending on the option selected, control branches to a series of subroutines. Foods are added to the new or existing list as a subroutine at lines 730–1030. The meat data are stored within the program in a two-dimensional string array, FOOD$ (row, column). The first row used, NF, indicates the number of foods in the file, and is incremented by one each time a new food is entered.

To simplify printing out options, code numbers instead of the actual names of the types of meat and cuts are stored in the array. The code number points to the element of TYPE$(n) and CUT$(n) appropriate to the item.

The first column of FOOD$ (row, column) stores a number from one to seven, which stands for the name of the meat represented by that element of the string array Type$(n). For example, TYPE$(1) = PORK, so if a cut of pork were to be entered, a string containing the numeral "1" would be inserted in FOOD$(NF,1).

The second column stores a string representation of the number that points to the CUT$(n) element equivalent to the cut selected. Chops would be the fourth element of CUT$(n), so to indicate pork chops, FOOD$(NF,2) would contain a four.

The third column contains the actual weight of the cut, input by the user to the nearest half pound. Because the weight is stored as a string, the user can enter a number only, such as a four, or the words four pounds. The numeral must come first, as the program looks for the VAL of FOOD$(n,3) when deleting a food from the list.

The date the food was frozen is entered into the fourth column (FOOD$(NF,4)). The format

MM/DD/YY must be used. That is, Aug. 4, 1981, should be entered as 08/04/81. No error trap is built in to catch improper input, and failure to conform to the standard MM/DD/YY will make it difficult to delete an inventory item later.

Entering new foods is fairly fast. Each of the TYPE$(n) and CUT$(n) options is presented to the screen in a pair of For... Next loops at lines 830–850 and lines 920–940 respectively. INKEY$ input will refuse any values other than those offered. Next, the user can input the weight of the meat. Weights to the closest pound are suggested, because of the needless complexity of converting pounds and ounces to decimal values. If two identical cuts of meat are entered on the same day, the user should enter slightly different weights to differentiate them. One could be four pounds, and the other 4.5 pounds. Mark these working weights on the package of frozen meat so the correct weight can be entered when the item is deleted from the inventory.

After the date is input, control reverts back to lines 770–800, where the information entered is transferred to the correct element of FOOD$(row, column). Then, the program returns to the menu.

You should remember to save current data to disk before exiting the program. By not having Save as an automatic step following the input of each new item of food, some time is saved whenever several items must be entered at one time. This will frequently be the case after a shopping trip, or when freezing a newly purchased side of meat.

In addition, a list of the current foods in inventory, including the new entries, can be requested—prior to the Save. If a mistake is discovered in the new entries, the user can delete the error (using menu option six) before the data is saved. Only one disk I/O operation need be performed at the end of the session.

Saving is performed at lines 1420–1490. The number of foods (NF) is first Printed to the disk file Foods, and then each element of FOOD$(row,column) is recorded. Because the disk I/O time difference is almost negligible between having the Row loop increment from one to 50 and from one to NF, I elected to use the former, simpler system for program clarity. Those who find it necessary to redimension FOOD$(row,col) larger than 50 rows may want to switch to the latter technique.

Accessing the File

The food file may be accessed three different ways. In lines 250–270, all the foods in the file are displayed. A loop from one to NF repeats until all foods are listed to the CRT. Pauses are built in every 12 foods by a check in line 290 that sends control to a wait subroutine at lines 400–430. If the loop counter (N4) can be evenly divided by 12(N4/12 = INT(N4/12)), then the user must hit any key to see the rest of the list.

If only one type of meat is to be listed, a subroutine at lines 450–710 checks FOOD$(N6,1) in line 590 to see if it matches the food type looked for (A$). If so, then the meat listing is printed to the CRT screen (lines 660–710), and a counter (CU) that keeps track of when 12 items have been found and displayed is incremented. The same wait subroutine is used.

Meats may also be listed by age (lines 1050–1400). The user inputs the current date as a

Program Listing

```
10 CLEAR 1000
20 SL$="/"
30 DIM TYPE$(7),CUT$(13),FOOD$(50,4)
40 GOSUB 1500
50 FOR N=1 TO 7:READ TYPE$(N):NEXT N
60 FOR N1=1 TO 13:READ CUT$(N1):NEXT N1
70 DATA PORK,BEEF,LAMB,FOWL,HAM,FISH,OTHER
80 DATA ROAST,ROAST-RUMP,ROAST-RIB,CHOP,STEAK-SIRLOIN,STEAK-CHUC
K,STEAK-ROUND,WHOLE,HALF,BREASTS,LEGS,GROUND,BRISKET
90 ' ****** MENU ******

100 CLS:PRINT:PRINT
110 PRINT " ENTER CHOICE :"
120 PRINT "        1.) SEE LIST OF ALL FOODS IN FREEZER"
130 PRINT "        2.) SEE LIST ONLY OF ONE TYPE FOOD IN FREEZER"

140 PRINT "        3.) ADD NEW FOODS TO FREEZER INVENTORY"
150 PRINT "        4.) SEE LIST OF FOODS BY AGE"
160 PRINT "        5.) SAVE INFORMATION INPUT THIS SESSION TO DIS
K"
170 PRINT "        6.) REMOVE AN ITEM FROM INVENTORY"
180 PRINT
190 PRINT "---->";
200 A$=INKEY$:IF A$="" GOTO 200
210 A=VAL(A$)
220 IF A<1 OR A>6 GOTO 200
230 ON A GOTO 240,440,720,1040,1410,1600
240 ' ****** LIST ALL FOODS IN FREEZER ******

250 CLS
260 PRINT "TYPE","CUT","WEIGHT","DATE"
270 PRINT
280 FOR N4=1 TO NF
290 IF INT(N4/12)=N4/12 GOSUB 390
300 PRINT TYPE$(VAL(FOOD$(N4,1))),
310 PRINT CUT$(VAL(FOOD$(N4,2))),
320 PRINT FOOD$(N4,3),
330 PRINT FOOD$(N4,4)
340 NEXT N4
350 PRINT
360 PRINT "HIT ANY KEY TO RETURN TO MENU"
370 IF INKEY$="" GOTO 370
380 GOTO 100
390 PRINT
400 PRINT "HIT ANY KEY TO SEE REST OF LIST"
410 IF INKEY$="" GOTO 410
420 CLS:PRINT
430 RETURN
440 ' ****** LIST ONLY ONE TYPE OF FOOD ******
450 CU=1
460 CLS:PRINT:PRINT
470 FOR N5=1 TO 6
480 PRINT N5;".) ";
490 PRINT TYPE$(N5)
500 NEXT N5
510 PRINT
520 PRINT "ENTER CHOICE";
530 A$=INKEY$:IF A$="" GOTO 530
540 A=VAL(A$)
550 IF A<1 OR A>6 GOTO 530
560 CLS:PRINT:PRINT
570 PRINT "ALL ";TYPE$(A);" LISTED BELOW:"
580 FOR N6=1 TO NF
590 IF FOOD$(N6,1)=A$ THEN GOSUB 660
600 NEXT N6
610 PRINT
620 IF CU=1 THEN PRINT "NO ";TYPE$(A);" FOUND IN FILE":PRINT
630 PRINT "HIT ANY KEY TO RETURN TO MENU"
640 IF INKEY$="" GOTO 640
650 GOTO 100
660 CU=CU+1
670 IF INT(CU/12)=CU/12 GOSUB 390
680 PRINT CUT$(VAL(FOOD$(N6,2))),
690 PRINT FOOD$(N6,3),
700 PRINT FOOD$(N6,4)
710 RETURN
720 ' ****** ADD NEW FOODS TO FREEZER INVENTORY ******

730 CLS:PRINT:PRINT
740 NF=NF+1
750 PRINT "FOOD TYPE OF NEW ENTRY :"
760 GOSUB 820
770 FOOD$(NF,1)=A$
780 FOOD$(NF,2)=A2$
790 FOOD$(NF,3)=A3$
800 FOOD$(NF,4)=A4$
810 GOTO 100
820 PRINT
830 FOR N2=1 TO 6
840 PRINT N2;".) ";TYPE$(N2)
850 NEXT N2
860 PRINT "ENTER CHOICE :";
870 A$=INKEY$:IF A$="" GOTO 870
880 A=VAL(A$)
890 IF A<1 OR A>6 GOTO 870
900 CLS
910 PRINT
920 FOR N3=1 TO 13
930 PRINT N3;".) ";CUT$(N3)
940 NEXT N3
950 PRINT
960 PRINT "ENTER CUT OR TYPE :";
970 INPUT A2$
980 A2=VAL(A2$)
990 IF A2<0 OR A2>13 GOTO 970
1000 CLS:PRINT:PRINT
1010 INPUT "ENTER WEIGHT TO CLOSEST POUND :";A3$
1020 INPUT "ENTER DATE FROZEN (MM/DD/YY)";A4$
1030 RETURN
1040 ' ****** SEE LIST OF FOODS BY AGE ******

1050 CLS:PRINT:PRINT
1060 INPUT "ENTER TODAY'S DATE : (MM/DD/YY) ";DATE$
1070 Y1$=RIGHT$(DATE$,2)
1080 M1$=LEFT$(DATE$,2)
```

Program continues

Program continued

```
1090 Y1=VAL(Y1$)
1100 M1=VAL(M1$)
1110 CLS:PRINT:PRINT
1120 FOR N7=1 TO 6
1130 TY$=Y1$
1140 TM=M1-N7
1150 IF TM<1 THEN TM=TM+12:TY=Y1-1:TY$=STR$(TY)
1160 PRINT
1170 PRINT "ALL FOODS ";N7;" MONTHS OLD LISTED BELOW"
1180 PRINT
1190 PRINT"TYPE","CUT","WEIGHT","DATE"
1200 PRINT
1210 CU=1
1220 FOR N8=1 TO NF
1230 Y$=RIGHT$(FOOD$(N8,4),2)
1240 M$=LEFT$(FOOD$(N8,4),2)
1250 IF VAL(Y$)=VAL(TY$) AND VAL(M$)=TM THEN GOSUB 1340
1260 NEXT N8
1270 PRINT "HIT ANY KEY FOR REST OF LIST"
1280 A$=INKEY$:IF A$="" GOTO 1280
1290 CLS:PRINT
1300 NEXT N7
1310 PRINT "HIT ENTER TO RETURN TO MENU"
1320 IF INKEY$="" GOTO 1320
1330 GOTO 100
1340 CU=CU+1
1350 IF CU/12=INT(CU/12) GOSUB 390
1360 PRINT TYPE$(VAL(FOOD$(N8,1))),
1370 PRINT CUT$(VAL(FOOD$(N8,2))),
1380 PRINT FOOD$(N8,3),
1390 PRINT FOOD$(N8,4)
1400 RETURN
1410 ' ****** SAVE FOODS TO DISK ******

1420 OPEN "O",1,"FOODS"
1430 PRINT #1,NF
1440 FOR ROW=1 TO 50
1450 FOR COL=1 TO 4
1460 PRINT #1,FOOD$(ROW,COL);",";
1470 NEXT COL,ROW
1480 CLOSE 1
1490 GOTO 100
1500 ' ***** LOAD FOODS FROM DISK ******

1510 OPEN "I",1,"FOODS"
1520 INPUT #1,NF
1530 FOR ROW=1 TO 50
1540 FOR COL=1 TO 4
1550 INPUT #1,FOOD$(ROW,COL)
1560 NEXT COL,ROW
1570 CLOSE 1
1580 RETURN
1590 '****** REMOVE FOOD FROM INVENTORY ******

1600 CLS:PRINT:PRINT
1610 PRINT "ENTER DATA ABOUT ITEM REMOVED FROM INVENTORY :"
1620 GOSUB 820
1630 FOR N9=1 TO NF
1640 IF VAL(FOOD$(N9,1))=VAL(A$) GOSUB 1670
1650 NEXT N9
1660 GOTO 100
1670 IF VAL(FOOD$(N9,2))=VAL(A2$) AND VAL(FOOD$(N9,3))=VAL(A3$)
GOTO 1690
1680 RETURN
1690 IF FOOD$(N9,4)<>A4$ THEN RETURN
1700 IF N9=NF THEN NF=NF-1:RETURN
1710 FOR M=1 TO 4
1720 FOOD$(N9,M)=FOOD$(NF,M)
1730 NEXT M
1740 NF=NF-1
1750 RETURN
```

MM/DD/YY string, which is parsed, and the values of the MM and YY compared with the values of the corresponding segments of each of the dates of the foods in the file. Target years (TY$) and target months (TM$) are determined for each of six months before the current month (lines 1130–1140). If a target month carries back into the previous year, then the value of TY$ is decremented by one, and the target month adjusted accordingly (line 1150). The loop prints out lists of meats one to six months old, month by month.

Deleting an Item from the File

As foods are consumed, they must be deleted from the file. This is accomplished by a routine at 1600–1750. The user enters data about the item to be removed (type, cut, weight and date frozen), and each element of FOOD$(row,column) is examined until a match is found. If the item to be deleted happens to be the last item in the file, (N9 = NF in line 1700), then NF is decremented by one, making that row in FOOD$ (row,column) available for new entries (which will wipe out the old ones, if still present). Effectively, the old entry does not exist, because loops accessing the data will not go beyond NF in their search.

More frequently, however, the item deleted will not be the last element of the array. Usually the oldest cuts of meat are consumed first. In this case, the item is deleted by taking the last set of data in the array, and moving it to the spot to be vacated. Then, NF is decremented by one. Effectively, this closes up the array by putting the last entry into the hole left by the last one deleted.

Enhancements

Freezer Inventory offers a number of opportunities for enhancement. Additional meat types and cuts can be tacked on by redimensioning the array TYPE$(n) and CUT$(n), and by enlarging the loops which display these choices to the user for input or retrieval. Use Step 2 to count off the display loops two numbers at a time, and change print lines, as in line 840 to read: 840 PRINT N2;".)": TYPE$(N2),N2 + 1;".)"; TYPE$(N2 + 1). This will allow displaying a greater number of choices on the screen at one time without scrolling off.

Frozen vegetables could also be added with a little imagination. Vegetable could be given as element 0 of TYPE$(n), and whenever FOOD$(row,1) was equal to zero then column 2 (cut) would be ignored, and, instead, a vegetable type would be input or retrieved from a new column in the array. ■

Coming Next Month

For November, *80 Microcomputing* takes a look at micros in business. We will cover topics ranging from real estate to the stock market.

Technical Editor Chris Brown has done an in-depth study of computers and the new white-collar militancy. In many businesses and federal agencies, there is a class of technical elite who have the power to bring these institutions to their knees by tampering with their computers. Some of these people have not hesitated to use this power.

News Editor Bert Latamore will give you the latest on electronic mail and its impact on the business sector. Who knows, maybe someday stamps and mailmen will be obsolete. Electronic mail is but one area that is revolutionizing communications.

Freelancer Theron Wieranga will show you how to use your TRS-80 to keep track of your personnel file. This program has already seen use in a school system and promises to be of interest to any businessman with a large number of employees.

Are you a realtor with a number of rental properties? Freelancer George Kwascha has a program that maintains rental records. Also of interest to realtors will be Leslie Sparks' real estate investment analysis program. ■

✓See List of Advertisers on page 386

✓See List of Advertisers on page 386

H & E COMPUTRONICS INC.

✓9

•••EVERYTHING FOR YOUR TRS-80™•••

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

★ *All orders processed within 24-Hours*
★ *30-Day money back guarantee on all Software*
★ *Add $3.00 for shipping in UPS Areas*
★ *Add $4.00 for C.O.D. or NON-UPS Areas*
★ *Add $5.00 outside U.S.A., Canada & Mexico*

SBSG SMALL BUSINESS SYSTEMS GROUP

COORDINATED BUSINESS SYSTEMS

VISA · AMERICAN EXPRESS · master charge

FACTS ABOUT THE S.B.S.G. BUSINESS PACKAGES

1. **S.B.S.G.** is a sophisticated Business Software System designed for the serious businessman.
2. Each of the **S.B.S.G. Business Modules** may be purchased separately...or you may purchase the entire coordinated business system.
3. Modules purchased separately do not coordinate with the General Ledger (although for the standard **S.B.S.G.** fee, the user may upgrade his individual modules for the coordinated system).
4. Foolproof, Step-By-Step procedures are supplied, planned and documented for the **First-Time Computer User.** All programs are self-explanatory, telling the user what is required at every step.
5. Programs are written in **BASIC** and the source code listing is supplied for those users who decide to modify the original system.
6. A complete users manual is supplied with each module.
7. Demo Data diskettes are supplied with sample data.
8. **S.B.S.G.** has an In-House staff that can answer questions and problems related to the proper use of the **S.B.S.G. Business System** (on the telephone or through the mail).
9. First-Time Computer Owners Note-Instructions are provided for entering state payroll withholding tables. There is an additional charge if you prefer to have **S.B.S.G. Programmers** insert the correct data.
10. Minimum system requirement is 2-drives to run any single module.
11. Minimum system requirement is 3-drives to run the coordinated business system (AR-AP-GL) or (AR-AP-GL with PAYROLL).
12. Minimum system requirement is 4-drives to run the extended coordinated system (AR-AP-GL-PR and INVENTORY/INVOICING).
13. The **A. OSBORNE & ASSOCIATES** business manuals are provided **FREE** with each order (they may be purchased separately at $20 per manual).
14. The INVENTORY and INVOICING modules are original programs written by **S.B.S.G.**
15. Each module can be purchased as independent modules to run on a 2 or more drive system except INVOICING.
16. Memory requirement is 48K for the MODEL-I and 64K for the MODEL-II.
17. All **S.B.S.G. BUSINESS SYSTEMS** may be upgraded up to 4-disk drives. No data is ever lost during an upgrade. There is a standard **S.B.S.G.** charge for all upgrades.

ACCOUNTS PAYABLE

The accounts payable system receives data concerning purchases from suppliers and produces checks in payment of outstanding invoices. In addition, it produces cash management reports. This system aids in tight financial control over all cash disbursements of the business. Several reports are available and supply information needed for the analysis of payments, expenses, purchases and cash requirements. All A/P data feeds General Ledger so that data is entered into the system just once. These programs were developed 5 years ago for the Wang micro-computer and have been tested in many environments since then. The package has been converted to the TRS-80™ and is now well documented, on-line, interactive micro-computer system with the capabilities of (or exceeding many larger systems).

CAPABILITIES:

★ menu driven; easy to use; full screen prompting and cursor control
★ invoice oriented; everything revolves around the invoice; handles new invoice or credit memo or debit memo
★ invoice information recorded; invoice #, description, buyer, check register #, invoice date, age date, amount of invoice, discount (in %), freight, tax ($), total payable
★ transaction print and file maintenance procedures insure accuracy
★ flexible check calculation procedure; allows checks to be calculated for a set of vendors-or-for specific vendors
★ program prints your checks; contiguous computer checks with your company letterhead can be purchased from SBSG
★ reports include (samples on back):
- open item listing/closed item listing - both detail and summary
- debit memo listing/credit memo listing
- aging
- check register report (to give an audit trail of checks printed)
- vendor listing and vendor activity (activity of the whole year)

★ fully linked to GENERAL LEDGER; each invoice can be distributed to as many as five (5) different GL accounts; system automatically posts to cash and A/P accounts

ACCOUNTS RECEIVABLE

The objective of a computerized A/R system is to prepare accurate and timeley monthly statements to credit customers. Management can generate information required to control the amount of credit extended and the collection of money owed in order to maximize profitable credit sales while minimizing losses from bad debts. The programs composing this system were developed 5 years ago, especially for small businesses using the Wang Microcomputer. They have been tested in many environments since then. Each module can be used stand alone or can feed General Ledger for a fully integrated system.

CAPABILITIES:

★ menu driven; easy to use; full screen prompting and cursor control
★ invoice oriented; invoices can be entered before ready for billing, when ready for billing, after billing or after paid
★ allows entry of new invoice, credit memo, debit memo, or change/delete invoice
★ allows for progress payment
★ transaction information includes:
- type of A/R transaction
- customer P.O. #
- description of P.O.
- billing date
- general ledger account number
- invoice amount
- shipping/transportation charges
- tax charges
- payment
- progress payment information
- transaction print & file maintenance procedures insure accuracy

★ customer statements printed; computer statements with your compay letterhead can be purchased from SBSG
★ reports include: (samples on back)
- listing of invoices not yet billed
- open items (unpaid invoices)
- closed items (paid invoices)
- aging

★ fully linked to General Ledger; will post to applicable accounts; debit A/R, credits account you specify

✓See List of Advertisers on page 386

✓See List of Advertisers on page 386

Tweak six additional characters out of Radio Shack's generator chip.

Soft Characters

Stephen D. Moulton
766 Biltmore Drive
Virginia Beach, VA 23454

Being a computer programmer both in profession and hobby, I was intrigued by a letter in *80 Input* ("Lowercase Access", November, 1980). The letter concerned those who had Radio Shack's lowercase modification installed, which included a brief program to POKE certain special characters into video memory. The letter also stated that these characters were otherwise inaccessible to Basic. If you run that program, you will see that a lowercase @ remains @, while lowercase arrows and underline keys produce spaces (see Program Listing 1).

At this point it occurred to me that, if the characters can be displayed with a series of POKE statements, they are obviously included in the Radio Shack hardware upgrade. The lack of accessibility to Basic must be caused by a deficiency in the software driver. I found that this is indeed the case, although I can not fathom why Radio Shack would deliberately ignore some of the capabilities of their character generator chip. At any rate, here is a fix for ULCBAS which will allow the display of six additional characters (£,{,¦,},ψ,■) under Basic control. In addition, the fix will repair the defective automatic memory protection feature so that the two POKE statements recommended by Radio Shack need not be entered prior to the execution of the driver program.

First, load and execute T-BUG and load ULCBAS using the L function. Now use the M function to make the changes listed in Table 1. The first two locations changed will repair the automatic memory protection feature so that your Basic program stack will not interfere with the driver software and cause a keyboard lockup. The remaining six changes add the additional characters to the driver program. Next, save the patched driver on tape using P 7000 7270 7000 ULC16K (this is the filename I use for my 16K system, but you can use any filename that you find convenient). At this point, I would turn off the TRS-80 just to make sure that any garbage I may have scattered through memory will be cleared, but a simple J 0000 should suffice.

To run the modified driver, first answer the memory size question with Enter (the program is self-protecting, right?). Next, type System, press Enter, type the filename you chose when you saved the program, and wait for the tape to load. Answer the *? prompt with

```
10 CLS
20 FOR X = 32 to 191
30 PRINT CHR$(32); CHR$(X);
40 NEXT
```

Program Listing

Change Location	From	To
709D	5D	F2
709E	70	7D for 16K system BD for 32K system FD for 48K system
7247	40	60
7262	20	7B
7263	20	7C
7264	20	7D
7265	20	7E
7266	20	7F

Table 1

/Enter and the familiar Ready prompt will appear. You now have the lowercase driver in protected memory, and ?MEM will yield 15046. To view your new character set, run the Program Listing 1 again. See the difference?

Now that we can access the additional characters from Basic, let us discuss how to use them in our programming. First, you will note that the @ key will always yield "@" when used alone and "£" when used with the shift key, regardless of the status of the Shift 0 toggle. I did this to make the @ key function like the other dual-symbol keys at the top of the keyboard. I also did not incorporate keyboard reversal for this key because £ will not be used as frequently as @ in normal programming applications. Note that the computer will not accept PRINT£ as PRINT@ (although the Exatron Stringy Floppy will accept £LOAD as @LOAD). Note also that Shift@ still functions to pause program execution. The next four special characters can not be input from the keyboard because the shifted arrow keys serve control functions. These symbols must be displayed using ?CHR$(123) through (126). The final symbol cannot be input from the keyboard because it is a shifted underline, and the TRS-80 has no underline key to shift! It can be displayed by ?CHR$(127).

There remains one additional note for owners of the Exatron Stringy Floppy. The lowercase driver software must be loaded and executed prior to the initialization of the ESF operating system (this applies to the original driver supplied by Radio Shack or to the one created in this article). After executing the driver program, the Stringy Floppy must be initialized using /12346 (*not* /12345) because the Exatron and lowercase driver debounce routines will interfere with one another.

So there you have it; 32 lowercase characters for the price of 26! ■

MULTI-USER OASIS HAS THE FEATURES PROS DEMAND. READ WHY.

(THEN COMPARE.)

NOW AVAILABLE FOR TRS80 MOD II

Computer experts (the pros) usually have big computer experience. That's why when they shop system software for Z80 micros, they look for the big system features they're used to. And that's why they like Multi-User OASIS. You will too.

DATA INTEGRITY: FILE & AUTOMATIC RECORD LOCKING

The biggest challenge for any multi-user system is co-ordinating requests from several users to change the sa... re...rd at the s...e.

Wi...out p... co...inatio... th... ...on a... problems o... ...curate or even destroyed da... ...n be staggering.

... Fil... ...atic ...ord l...king fe...ures ...olve ...ese ...oblems.

For ...xample: normally a... users can view a particular record at the same time. But, if that record is being updated by one user, automatic record locking will deny all other users access to the record until the up-date is completed. So records are always accurate, up-to-date and integrity is assured.

Pros demand file & automatic record locking. OASIS has it.

SYSTEM SECURITY: LOGON, PASSWORD & USER ACCOUNTING

Controlling who gets on your system and what they do once they're on it is the essence of system security.

Without this control, unauthorized users could access your programs and data and do what they like. A frightening prospect isn't it?

And multi-users can multiply the problem.

But with the Logon Password and Privi... Level feat... ...Mu...ser OAS... a s...m ...an... ...spec... ...user whic... ...s and files ... ...essed — a... for what pur...e.

Security ...fur... enhanced ... ...se... A...counting ... ...that l...s you keep ... history of which user has been ...ged on, when and for how long.

Pros insist on these security features. OASIS has them.

EFFICIENCY: RE-ENTRANT BASIC

A multi-user system is often not even practical on computers limited to 64K memory.

OASIS Re-entrant BASIC makes it practical.

How?

Because all users use a single run-time BASIC module, to execute their compiled programs, less memory is needed. Even if you have more than 64K, your pay-off is cost saving and more efficient use of all the memor... you ha... available ... ...it se...ices ...ore ...se...

...d ...e a p... ...ture? It i... A... O... ...as it.

AND LOTS MORE...

M... ...er O... supports a... many ...s 16 ...inals ... can ...n in as little as ... ...ory. Or, with ...ank switching, as much as 784K.

Multi-Tasking lets each user run more than one job at the same time.

And there's our BASIC — a compiler, interpreter and debugger all in one. An OASIS exclusive.

Still more: Editor; Hard & Floppy Disk Support; Keyed (ISAM), Direct & Sequential Files; Mail-Box; Scheduler; Spooler; all from OASIS.

Our documentation is recognized as some of the best, most extensive, in the industry. And, of course, there's plenty of application software.

Put it all together and it's easy to see why the real pros like OASIS. Join them. Send your order today.

OASIS IS AVAILABLE FOR

SYSTEMS: Altos; Compucorp; Cromemco; Delta Products; Digital Group; Digital Microsystems; Dynabyte; Godbout; ... Index; Intersystems; North... ...yx... SD Systems; TRS 80 M... ...; Vec... Graphic; Vorime...

CONTRO... ...S: Bell ...trols; Car...; Corvus; Ko...; Mic... ...n; Micro...s; ...rbell; Tele... ...kerto... ...Co...

...e for comp... ...ree Application ...oftware Directory

PLEASE SEND ME:

Product	Price with Manual	Manual Only
OPERATING SYSTEM (Includes: EXEC Language; File Management; User Accounting; Device Drivers; Print Spooler; General Text Editor; etc.) SINGLE-USER	$150	$17.50
MULTI-USER	350	17.50
BASIC COMPILER/ INTERPRETER/DEBUGGER	100	15.00
RE-ENTRANT BASIC COMPILER/INTERPRETER/ DEBUGGER	150	15.00
DEVELOPMENT PACKAGE (Macro Assembler; Linkage Editor; Debugger)	150	25.00
TEXT EDITOR & SCRIPT PROCESSOR	150	15.00
DIAGNOSTIC & CONVERSION UTILITIES (Memory Test, Assembly Language; Converters, File Recovery, Disk Test; File Copy from other OS; etc.)	100	15.00
COMMUNICATIONS PACKAGE (Terminal Emulator; File Send & Receive)	100	15.00
PACKAGE PRICE (All of Above) SINGLE-USER	500	60.00
MULTI-USER	850	60.00
FILE SORT	100	15.00
COBOL-ANSI '74	750	35.00

Order OASIS from:

Phase One Systems, Inc.
7700 Edgewater Drive, Suite 830
Oakland, CA 94621
Telephone (415) 562-8085
TWX 910-366-7139

NAME ____________
STREET (NO BOX #) ________
CITY ____________
STATE ________ ZIP ________

AMOUNT $ ________
(Attach system description; add $3 for shipping; California residents add sales tax)

☐ Check enclosed ☐ VISA
☐ UPS C.O.D. ☐ Mastercharge

Card Number ____________
Expiration Date ____________
Signature ____________

OASIS

MAKES MICROS RUN LIKE MINIS

A TRSDOS to CP/M conversion program.

Crossdos

Jack Means
1612 East Oakland
Bloomington, IL 61701

Digital Research's CPM is neither the most sophisticated nor the most convenient operating system available for the TRS-80. However, the large number of programming languages and quality business packages available for TRS-80 CPM made me more than willing to order it.

Immediately after receiving it, I was faced with two problems. First, the text editor provided with CPM (ED) was too limited and cumbersome to suit my needs. Second, I needed some way to transfer programs and data files from TRSDOS (or NEWDOS) to CPM.

A Solution

Program Listing 1 solves both problems. I can now create a program under TRSDOS using virtually any text editor (Electric Pencil, Scripsit, Electric Secretary, even the TRS-80 Basic editor), and transfer the file to CPM using the attached program. In addition, the program allows me to transfer any sequential data set from TRSDOS to CPM. The program can handle random files, but they must first be converted to sequential data sets.

To use the program, load the file and type "Run". The system will immediately display: Enter input file name:.

At this point, enter the name of the TRSDOS file to be transferred. If the file is a Basic program created with the TRS-80 Basic editor, it must have been saved with the ASCII option (i.e., SAVE "MAILLST/BAS",A). The next video display will be: Enter name for output CPM file:.

Enter the name which you would like to have assigned to the file after it is transferred to CPM. Remember, if you plan to use the file as input to the CBasic compiler, it must have the suffix of .BAS (i.e., MAILLST.BAS).

Enter CPM Drive (A,B,C,D) for output:

Enter the CPM drive name to which the output data set should be written. Remember that CPM uses A, B, C, D instead of 1, 2, 3, 4.

The system will now display the word Converting: and display each text line as it is read into the system. When the entire file has been read into memory, the system will display something similar to the following:

```
10 '  ********************************************************
20 '  * * * * *          T R S 8 0  -  C P M           * * * * *
30 '  * * * * *     T R A N S F E R   P R O G R A M     * * * * *
40 '  * * * * *            C O P Y R I G H T            * * * * *
50 '  * * * * *               JACK MEANS                * * * * *
60 '  * * * * *           1612 EAST OAKLAND             * * * * *
70 '  * * * * *              BLOOMINGTON                * * * * *
80 '  * * * * *               ILLINOIS                  * * * * *
90 '  * * * * *          ALL RIGHTS RESERVED            * * * * *
100 ' ********************************************************
110 CLEAR 500 : ML=65535 : CLS : DEFUSR1=&H0000
120 LINEINPUT "ENTER TRSDOS INPUT FILE NAME: ";N$
130 LINEINPUT "ENTER NAME FOR OUTPUT CPM FILE: ";C$
140 LINEINPUT "ENTER CPM DRIVE (A,B,C,D) FOR OUTPUT: ";D$
150 IF D$ < "A" OR D$ > "D" THEN GOTO 140
160 OPEN "I",1,N$
170 LINEINPUT#1,I$
180 IF EOF(1) THEN GOTO 280
190 L=LEN(I$) : IF L=0 THEN I$=" "
200 PRINT@448,"CONVERTING:" : PRINT@512,I$;CHR$(31)
210 FOR I=1 TO L
220 A=ASC(MID$(I$,I,1))
230 POKE (-1*(ML-32767)),A : ML=ML-1
240 NEXT I
250 POKE(-1*(ML-32767)),13 : ML=ML-1
260 POKE(-1*(ML-32767)),10 : ML=ML-1
270 IF ML < 33535 THEN GOTO 350 ELSE GOTO 170
280 L=LEN(I$) : IF L=0 THEN GOTO 330
290 FOR I=1 TO L
300 A=ASC(MID$(I$,I,1))
310 POKE(-1*(ML-32767)),A : ML=ML-1
320 NEXT I
330 POKE(-1*(ML-32767)),13 : ML=ML-1
340 POKE(-1*(ML-32767)),10 : CLS
350 PRINT "FILE ";N$;" HAS BEEN WRITTEN IN MEMORY."
360 PRINT "PLEASE COPY THE FOLLOWING STATEMENTS EXACTLY:":PRINT
370 A=32768+(65535-ML) : Z=16 : GOSUB 440 : GOTO 390
380 N=N-1 : IF N=0 THEN GOTO 430
390 IF B(N)=>10 THEN GOTO 420
400 A1=B(N)+48
410 ML$=ML$+CHR$(A1) : GOTO 380
420 A1=B(N)+55 : GOTO 410
430 GOTO 480
440 N=N+1 : B=INT(A/Z) : C=A-(B*Z) : B(N)=INT(C+.5)
450 B(N)=INT(C+.5) : A=B
460 IF A>0 THEN GOTO 440
470 RETURN
480 PRINT "M8000,";ML$;",4300"
490 Z=INT((65535-ML)/256+1)
500 PRINT "SAVE";Z;D$;":";C$:PRINT
510 LINEINPUT "PRESS ENTER WHEN YOU HAVE COPIED THE STATEMENTS "
;X$
520 CLS : PRINT "REMOVE ALL TRSDOS DISKS."
530 PRINT "INSERT THE CPM SYSTEM DISKETTE IN DRIVE A. "
540 PRINT "INSERT A CPM FORMATED DISKETTE IN DRIVE ";D$ : PRINT
550 LINEINPUT "PRESS ENTER WHEN THE CPM DISKS ARE READY ";I$
560 X=USR1(0)
```

Program Listing

File MAILLST/BAS has been written In memory. Please copy the following statements exactly:

M8000,875D,4300
SAVE 8 B:MAILLST.BAS
Press Enter when you have copied the statements.

Once the Enter key has been pressed, the screen will clear and the following will be displayed:

Remove all TRSDOS disks.
Insert the CPM system disk in drive A.
Insert a CPM formatted disk in drive B.
Press Enter when the CPM disks are ready.

At this point, remove all TRSDOS disks and place the CPM System disk in drive A (TRSDOS 0) and the disk for the output in the drive indicated. If the data set is to be written on the System disk, the second statement can be ignored.

Once the CPM disks have been loaded, press Enter. The CPM operating system will be loaded and the prompt A> will be displayed. At this point you should enter: DDT.

The system will then display the following: "-".

Next, enter the first of the two statements you copied earlier (i.e., M8000,875D,4300). The system will respond with: "-".

Now press the Break key. This will cause the CPM to reboot. As soon as the A> prompt appears, enter the second of the two statements copied earlier (i.e., SAVE 8 B:MAILLST.BAS).

A copy of the transferred file is now in the drive specified. It can be edited by ED as if it had been created on the CPM system. However, the last line in the file will usually contain garbage. To eliminate the line simply use the ED kill (K) command.

As written, the system will transfer just over 32,000 bytes at a time. If your file is larger than this, it can be segmented and transmitted as individual files. Once on the CPM system, you can again merge the segments. ■

✓See List of Advertisers on page 386

Is anybody there. . . does anybody care?

Adventures in Modemland

Fred Blechman
7217 Bernadine Ave.
Canoga Park, CA 91307

So you're a new Model III owner—or you're thinking of getting a Model III. One thing you want to know is, "Are there a lot of programs I can use?" The answer is an emphatic *yes!* The vast majority of Basic programs written for the TRS-80 Model I will run without modification on the Model III—but *not* all of them.

But first let's get some confusing terminology straight. Radio Shack has chosen to use the TRS-80 name on all their recent computers (Model I, Model II, Model III, Pocket Computer, Color Computer and Videotex Terminal). However, the Basic interpreters for these computers are not the same; some programs require more memory than others; and some programs are written for disk-based systems, others strictly for cassette.

In both the Model I and the Model III the less-sophisticated Basic is called Level I. They seem to be 100 percent compatible, with the added feature Model III Level I includes—printer commands not found in Model I Level I.

Model I Level II Basic is largely compatible with Model III Basic. This Basic is not called Level II with the Model III, possibly to avoid confusion with Model II Basic which is quite different.

I have a cassette-based Model I Level II 16K unit for which I've written and sold various programs (Amway Products Distributor programs and Small Home-Business programs). I also have 22 programs in a new Hayden book (*Programs for Beginners on the TRS-80*). Since all these programs were written for Model I, I wondered if they would run on Model III.

Real Trouble

I bought the *TRS-80 Model III Operation and Basic Language Reference Manual* (Radio Shack Catalog Number 26-2112, $5.95) and read it cover to cover. This convinced me the Model III is more than just a repackaging of the Model I, since it has so many additional features.

Since all my programs are in Level II Basic, I didn't concern myself with Level I, but looked for information on the programming compatibility with Model III Basic. In the 270-page Model III manual, less than one page is devoted to Model I to Model III conversion hints! What particularly grabbed my attention were the differences in available memory (less in Model III) and the slightly different character sets. I could see I was in trouble, but later found the trouble was worse than I thought. Many Level II Model III differences are not mentioned in the manual!

It became obvious that the only certain way to find out how my programs would operate with a Model III would be to run them on one. Armed with my Level II cassettes, my own recorder and a loading meter (to reduce tape loading problems), I went to the local Radio Shack Computer Center and explained my dilemma to the store manager. He could not have been more cooperative.

There were about six Model Is and six Model IIIs set up for their classroom, and he said I could use them as long as I didn't interfere with his classes.

I found a 16K Model III Basic machine and immediately ran into the first thing not mentioned in the Model III manual. It seems Radio Shack recently changed the DIN five-pin plug on the cassette recorder-to-computer cable from one with a thick plastic sleeve to one with a thin metal one. The old type plastic sleeve DIN plug will not fit the Model III socket! You must have the new Radio Shack cassette recorder cable, now supplied with the CTR-80A recorder also available separately (Catalog Number AW-2577 for $6.95). The new cable fits both Model I and Model III.

The manager loaned me a new cable. We plugged everything in, turned on the Model III and "CASS?" appeared on the screen; this was the computer asking me what cassette speed to use. Model III Basic uses 500 baud for data recording; 1500 or 500 baud for cassette program loading or saving. Since Level II uses 500 baud only and my cassettes were recorded in Level II, I typed and entered L for low cassette speed. The default value (pressing Enter without typing anything) will result in a high cassette speed and Level II tapes will not load!

The first program I tried was my "*Amway Order Verification*," which loads into about 10,400 bytes of memory: no problem—

until I tried to run it. While dimensioning a large array—something I had no problem doing with my Model I 16K—the Model III told me it was out of memory. Model III Basic has less available memory.

A Definite Lack

How much less? That depends on your Model I. The Model III has 258 bytes less user memory than the older versions of Model I Level II, 256 bytes less than later versions. When you type and enter: ?MEM or PRINT MEM on the Model I, you get 15572 (older Model I) or 15570 (new Model I) for Level II 16K. On Model III 16K Basic you get 15314. This is the best way to determine no extra memory has been reserved for Clear, arrays or upper-memory programs. (Actually, 50 bytes are automatically cleared on power-up for string handling. To prove this, type and enter CLEAR0 and then ?MEM and you'll find 15364 as the Model III Basic total free memory, but you won't be able to handle any strings at all. Type and enter CLEAR50 and you're back to power-up configuration.)

Fortunately in this program, the entire array was not really required. I reduced the array size and the order verification program ran beautifully. In another version of this program, which needed the maximum array size, I removed some remark lines to free enough memory.

A Mystery Revealed

Next I loaded my "Amway Monthly Gross Profit" program—plenty of memory to spare, no problem there. However, right at the beginning of the program I PEEKed at memory location 14312 to determine if the printer was ready. A number in this memory location indicates the printer status in the Model I. The printer status is at the same memory location in Model III Basic. Unfortunately, the value of the number at this memory location is not the same in the Model I and Model III.

The solution to this problem is to PEEK at location 14312 with your printer on and ready and note the number for that particular printer. Be aware that different printers may give a different value at 14312—especially if you are using a serial printer with a printer-driver program in memory. The printer-ready number at 14312 can then be used by the program to determine if the printer is ready. If not, the program can branch around LPRINT commands.

Also, it's not generally known (and not in the TRS-80 manuals) that with Level II Basic you can punch in POKE16423,4 to make your program transparent to LPRINT statements. This also works with Model III Basic.

However, to restore program recognition of LPRINT, you must insert POKE16423,5 for Model I or POKE16423,3 for Model III. Do not put in POKE16423,5 on the Model III or you will lock the computer in Disk Basic and will be able to recover only by turning the computer off (thus losing your program) and then turning it back on after 10 seconds while also pressing the Break key. I found this out from the store manager when I unintentionally locked the computer center's Model III in Disk Basic and thought I had destroyed it!

With Renewed Courage

Regaining my courage, I loaded and ran five other Amway programs and found no other changes necessary. Then I turned to the Small Home-Business Package designed for Level II 16K with printer. The first program, "Speed-Letter," a simple text generator, bombed out with the message ?OM ERROR in 130. Line 130 of that program is CLEAR9400, to preserve lots of string space for text. The fewer bytes of free memory in Model III Basic caused this to happen, so I changed line 130 to CLEAR9000 to reduce the memory requirement. This allowed it to run in Model III Basic, but also reduced the length of text permitted in using the program.

That program also uses an

✓See List of Advertisers on page 386

up arrow to indicate the end of text. The Model III video display character set is not identical to the Model I. In particular, there are 96 special additional characters, but no up, down, left or right arrows! It seems incredible these were not included in the new 96 characters. The up arrow on the Model I keyboard produced an up arrow on the screen. This could also be generated using the command CHR$(91). The Model III keyboard still has an up arrow, but this key (or CHR$(91)) produces a left bracket instead of an up arrow. I didn't change my program but instead, tolerate a left bracket instead of an up arrow on the screen. This could really be a problem in a program that used arrows to indicate direction, such as some game programs.

Another Program Bombs

My "12-Column Ledger" and "3-Across Mailing Labels" programs loaded and ran with no changes.

However, I ran into two Model III Basic problems with my "Telephone Dialer/Timer" program. This program automatically dials a telephone, using up to 500 names and phone numbers stored in memory. The dialing is accomplished by using the internal TRS-80 relay to trigger an external buffer relay whose contacts interrupt the phone circuit. The recorder cable subminiature plug (which normally goes to the recorder remote jack) is directly connected to the internal TRS-80 relay contacts. The OUT255,4 command on the Model I closes this relay's contacts, and OUT255,0 is used to open the relay contacts. This does not work on the Model III, so I got help from the manager again. He and the Computer Center lead technician went through some Model III schematics and told me to try OUT236,2 to close the relay and OUT236,0 to open it. It worked!

The timing portion of this program, used to show elapsed time in seconds and minutes for cost calculation, was running too fast. The Model III runs at 2.02752 MHz, while the Model I runs at 1.774 MHz. Consequently, any For...Next loops used for timing need to be modified. Unfortunately, although the Model III internal clock speed is approximately 14.3 percent faster than the Model I, the Basic speed of operation is not directly related by the same percentage. In this program I found a FOR X = 1 TO 266:NEXT loop had to be changed to FOR X = 1 TO 285:NEXT (only a seven percent increase in counts) to retain one second for a completed loop count. It becomes a matter of trial and error and will vary with individual machines and room temperature. The circuits are crystal controlled, with no real attempt at extreme accuracy or long-term temperature stability, so don't expect your TRS-80 timing loops to have even the accuracy of your digital wristwatch.

I ran into the same thing in my "Telephone Toll-Charge" program, where a similar For...Next loop count had to be increased from 263 to 325 (an almost 24 percent increase in counts) to maintain a nominal one-second completed loop. Several things affect the running speed of For...Next loops: the location of the loop within the program; whether the variable is an integer, single or double precision; and to some degree, the number of other variables that precede this loop variable in the program.

Next, I decided to see how the 22 programs I wrote for the Hayden book would run in Model III Basic. Most of these programs were originally written in Level I 4K Basic, then changed to Level II. Since the book features line-by-line explanations of each program, I wanted to see if any rewriting would be necessary. I found that except for the numbers in some timing loops, no other program changes were necessary. So, Model III runs many simple Model I programs with little or no change.

The book also contains several appendices, and I checked their validity for Model III Basic. A two-line keyboard debounce program, needed in

Level II to prevent multiple entries when pressing a key only once, is not needed for the Model III, since the keyboard does not have this problem. My Audio/Visual Control Box, for reliably loading cassette programs and controlling recorder operation, worked perfectly with the Model III without any changes. Simple subroutines for printing the screen display on a printer, with and without graphics, worked perfectly. But I ran into trouble using a simple two-line Basic program merging technique.

Another Discovery

The Model III manual memory map (page 12/23 of the operation section) shows 17129 as the beginning of user memory. However, decimal memory locations 16548 and 16549 are the beginning-of-program pointer. The numbers in these two locations are the least and most significant bits, in decimal form, of an eight-digit byte giving the decimal location of the beginning of the Basic program area.

For Model I Level II, location 16548 shows a 233 and location 16549 has a 66. Multiply 66 by 256 and add 233 and you get 17129. However, the Model III on power-up shows a 67 in location 16549 and 233 in 16548. Since 67 times 256 plus 233 equals 17385, it seems the beginning of Basic programs is 17385—exactly 256 bytes higher than in the Model I.

Apparently this is where the Model III lost 256 bytes. I confirmed this with some PEEKs at a Basic program in memory and by successfully merging programs using the technique in the book, revised to reflect the new beginning-of-program location.

While you may wonder if this is important, I can tell you this change will raise havoc with any System tapes you have that load into low memory. There are various utility programs that do this. They won't work unless modified and for most of them this means "back to the drawing board," since they are written in machine language, not Basic.

Another item in my book appendix needed revision. I have a one-liner that makes two cassette copies with a two-second gap between them. I put this at the end of all my programs as the last line:

```
CSAVE"1":OUT255,4:FOR I=1 TO
2000:NEXT:CSAVE"1"
```

See the problem? OUT255,4 must be changed to OUT236,2 for Model III. With this as the last program line, you merely type run line number and Enter and the computer dumps two copies on tape.

One of my programs uses a WEB International T-Beep, a device that beeps on command from the computer. This tells you when the computer has finished a task (sort, search, load, save or a program error). It uses a simple command and time-loop for operation from the cassette auxiliary cable. The command OUT255,1 causes the voltage at the computer's gray miniature phone plug to go from its normal .45 volts dc to .9 volts dc. This is enough to make the T-Beep sound. The command OUT 255,0 returns the voltage to .45 volts. I checked this out on the Model III and these commands have not been changed.

There are many other subtle differences between Level II and Model III Basic in various ROM addresses that can crash your programs written for Level II in machine code. Also, Basic POKEs and PEEKs will have to be carefully checked. If your Level II program crashes in Model III—especially with a ?CASS or the memory size query on the screen—you probably POKEd into the wrong location. ■

Difference	Level II Basic 16K	Model III Basic 16K
Cassette Cable*	Plastic DIN Plug	Metal DIN Plug (R/S#AW-2577)
Less Memory (256 or 258 Bytes)	?MEM = 15772 or 15770	?MEM = 15364
Printer Status* (Same Location, Different Value)	?PEEK (14312) (Note Value)	?PEEK (14312) (Note Value)
LPRINT*	POKE 16423,4 to Disable POKE 16423,5 to Enable	POKE 16423,4 to Disable POKE 16423,3 to Enable
CHR$(91)	↑	[
Arrows	↑ → ↓ ←	None
Speed (Timing Loops)	1.774 MHz	2.02752 MHz
Internal Relay*	Out 255,4 to Close Out 255,0 to Open	Out 236,2 to Close Out 236,0 to Open
Beginning of Basic Program*	17129 (42E9 Hex)	17385 (43E9 Hex)

*Not covered (or incorrect!) in Model III manual.

Table 1. Some Level II/Model III Basic Differences

See List of Advertisers on page 386

AMBER EAZE™

EYE FATIGUE— Much discussion has taken place regarding screen Color and contrast. Extensive research has produced a breakthrough in the prevention of Operator Eye Fatigue.

Amber has been found to be much more effective in preventing eye fatigue than any other color. Put an Amber Eaze Filter on your TRS-80™ or any other computer monitor. Contact one of the dealers below.

Model I $19.95
Model II & III. . $24.95
Other Computers Call

The Copyright Kit

"How to Copyright your Computer Software"

A self-instruction booklet on copyrighting the computer programs you write. INCLUDES: Step by step instructions, sample forms, as well as discussions of copyrights, patients and trade secrets, your rights secured by copyrights, legal remedies upon infringements, material not copyrightable and MUCH MORE! Written by Attorneys. Published by National Attorneys Publications Inc. and distributed exclusively through B.T. Enterprises and available from the dealers below. IF YOU EVER WROTE A COMPUTER PROGRAM, YOU NEED THIS BOOK!!

Available At

Breeze Computing
P.O. Box 1013
Berkley, MI 48072
313-288-9422

Computer Magic Inc.
115 Weltshire Ave.
Louisville, KY 40207
502-893-9334

All Systems Go
105 W. Plant St.
Winter Garden, FL 32781
305-877-2830

CPU Computer Corp.
175 Main Street
Charleston, MA 02129

Programs Unlimited
20 Jericho Tpke.
Jericho, NY 11753
800-645-6038

Son John Enterprises
P.O. Box 1671
Freemont, CA 94538
415-651-4147

Bob's T.V.
93-15 86th Drive
Woodhaven, NY 11421
212-441-2807

Computer Serv. of Danbury
P.O. Box 993
Danbury, Ct. 06810
203-744-5516

Contemporary Comm Corp.
5582 Coral Way
Haslett, MI 48840
517-339-1028

Stevens Radio Shack
562 Nutt Road
Phoenixville, PA 19460
800-345-6279

Soft Sector Marketing
6250 Middle Belt Rd.
Garden City, MI 48135
313-425-4020

Omni Systems Associates
P.O. Box 632
W. Caldwell, NJ 07006
201-226-9185

BT ENTERPRISES
171 Hawkins Road ✓122
Centereach, New York 11720
(516) 981-8568 (Voice)
(516) 588-5836 (Data)
MNET-70331, 105

Dealer Inquiries Welcome
Add $2.00 S & H
NYS res. add appr. tax

VISA MasterCard

✓See List of Advertisers on page 386

Harmony and multiplicity.

Multi-Statement

Dan Keen and Dave Dischert
Custom Tailored
Computer Software
RD # 1, Box 432,
State Highway 83
Cape May Court House, NJ
08210

One of the best features of Basic programming is that once you become good at it, there are more conquests to make as you discover new methods to use so your programs look neater, execute faster, and use less memory space. We will examine some tricks that will help you round off those rough edges.

It never occurs to most programmers that data statements can exist in harmony on multiple statement lines. The Model I owner's manual gives the impression that data lines should be tucked away by themselves at the bottom of the program. But, both of these lines are valid:

```
10 DATA 23,25,89,100: FORX-1 to 4: READ A(X): NEXT
10 FORX = 1 to 4: READ A(X): NEXT: DATA 23,25,89,100
```

Don't hold back on the length of multiple statement lines, pack them full. Multiple statements save bytes and decrease program run time.

```
10 CLS
20 PRINT@256,"<A>DD":PRINT"<R>EPORT":PRINT"<D>ELETE":PRINT"<S>ET
 UP"
30 A$=INKEY$:IFA$="A"THEN100ELSEIFA$="R"THENRUN"REPORT/BAS"ELSEI
FA$="D"THENRUN"DELETE/BAS"ELSEIFA$="S"THENGOTO200
40 FORX=1TO50:NEXT:GOSUB60:FORX=1TO100:NEXT:GOSUB70
50 GOTO30
60 PRINT@257,"A";:PRINT@321,"R";:PRINT@385,"D";:PRINT@449,"S";:R
ETURN
70 PRINT@257," ";:PRINT@321," ";:PRINT@385," ";:PRINT@449," ";:R
ETURN
```

Program Listing 1.

```
10 ONERRORGOTO100:DEFUSR=123456
20 REM
        OTHER INSTRUCTIONS GO HERE
        FOR DISK
80 GOTO200
100 POKE16526,L. S. B. :POKE16527,M. S. B.
        OTHER INSTRUCTIONS GO HERE
        FOR TAPE
200 ONERRORGOTO0:REM
        PROGRAM STARTS HERE
210 X=USR(Q)
```

Program Listing 2.

Program Listing 1 shows a sample menu. An option is to be selected. Check out line 30—it's packed, but very efficient. The book didn't tell you that it's possible to use more than one If . . . Then . . . Else statement on a line, but you can use as many as you want. Just keep track so each If has a Then, and an Else if needed.

There are two other interesting points to be made about that short program in Listing 1. First, if some options are selected, program execution may branch to another section within the program (by using GOTOs), or it may run an entirely different program. All the command words listed in chapter two in your Level II owner's manual can be used. The only exception is CONT: It's not that the machine doesn't recognize that word, it just doesn't know what to continue.

Some commands don't return you to your Basic program after they execute, such as CLOAD, List and System.

Experiment deleting lines or even causing New to execute if the operator indicates he's done with this program.

The second point of interest is dynamic flashing of the characters to be selected. As you can see, the program prompts you to respond with A, R, D or S. Between checking the keyboard via INKEY$, these letters are blinked on and off. Since the Model I doesn't have a flashing cursor, this animated input lets the operator know that the computer is ready and awaiting further instructions.

Potpourri

You can use a PEEK statement as the value part of a POKE statement. This line takes the character on the upper-most left side of the screen and displays it near the center of the screen:

```
10 POKE 15850, PEEK (15360)
```

Variables with a declaration character after them may or may not be the same as the plain variable. For example, if you use the variable SR%, don't expect SR to have the same value. We've run into trouble assuming that the two would always be equal.

Auto Date Routine

If you have a program which uses the current day's date within it, put the auto date check routine at the top of Program Listing 4. The routine determines if the date has been set. If the date has not been set, it will ask you to Enter the date. Should the date already be set, then it will display the date and ask Is Date O.K.? Execute these lines before a clear statement in your program so none of the variables here will interfere with those used later on.

The routine simply PEEKs the address where the day is stored. If the value there is zero, the date has not been set.

The For . . . Next Trick

Check out Program Listing 3. See the NEXT A in lines 50, 60,

70, 80, and 100? You could rewrite the program by changing all of those NEXT As to GOTO 20 and delete line 10. But the program as it stands takes up fewer bytes than would the GOTO version even with that extra line deleted!

This method is a convenient way of returning to the same point in a program. Perhaps the returned point could set up something on the screen, or reset some variable. By stepping zero, the loop never runs out of counts.

The For . . . Next Step 0 technique is not necessarily a great advantage over GOTOs and GOSUBs, but it does use less memory.

A Disk Basic Trick

Here's a trick for Disk Basic programmers. We had a program where we were cramped for disk space and each byte was important. The program must store many dates, both the month and the day. The trick is to take the dates and manipulate them so they take up less disk space.

Since there is no need to store a slash (/), just create one number from the two. Then you can use MKI$ and store the number as an integer. Upon retrieving the number, do the reverse. Add a slash and split the number in two. This saves bytes over storing the date in its entered form:

```
10 INPUT"ENTER MONTH AND DAY
(MM/DD) ";DA$
20 DA = VAL(LEFT$,2) = RIGHT$(DA,2))
30 LSET(field variable) = MKI$(DA)

To restore:
10 B$ = STR$(CVI(field variable))
2 0
B$ = MID$,(B$1,2) + "/" + RIGHT$(B$,2)
30 PRINT B$
```

Game Tips

When writing a game or a children's program which utilizes an INKEY$, use a dummy INKEY just prior to the one you want. This picks up any stray keys that may have been pressed before you want a response from the operator. Here's how it could be set up:

```
10 IK$ = INKEY$
20 PRINT"SELECT A LETTER"
30 A$ = INKEY$: IF A$ = ""THEN 30
40 -----PROGRAM TEXT-----
```

The variable IK$ will never be used anywhere in the program. It simply prevents previous key entries from getting through to line 30, where the real variable, A$, is quizzed. Let's go one step better than INKEY$ for getting input from the operator.

By PEEKing memory location 15350, you can find a value returned by the keyboard. This method has two advantages over INKEY$. A$ = INKEY$ won't hold the value if it is in a loop. The next time that statement is hit, A$ would change to become whatever is there.

Also, PEEK will detect if more than one key is being depressed. This is great for checking to see if the arrows are being held down. The up arrow returns an eight, down arrow 16, left arrow 32, and the right arrow 64. If you hit the left and up arrows together, the memory location holds a 40, which is the addition of 32 and eight. Furthermore, holding down a key will enable you to design continuous firing into your game.

The negative aspect of this procedure is that we're only looking at one byte of the keyboard return, so other keys will return the same value. Both the left arrow and the letter T, for example, return a value of 32 but it generally serves the purpose for most games.

Disk and Level II USR Calls

Our final tip deals with the use of the USR command. Writing a program with a USR call can get you into trouble if you want it compatible with both Level II and Disk Basic. Since Disk Basic supports more than one USR call, it must be set up a little differently. If the program has only one USR call, an error handling routine can trap it and branch program execution to an appropriate section.

To tell the computer where to go when a USR call is encountered, Disk Basic requires the address to be set up by DEFUSR-address. Level II Basic has the address value POKEd into memory locations 16526 (least significant byte) and 16527 (most significant byte).

A DEFUSR statement would give an L3 error if it were run under Level II. Take a look at Program Listing 2. If this was run in Level II, line 10 would generate an error but because of the On Error statement the program won't go out on error, but rather will branch to line 100 where it can be set up for Level II operation. Similarly, if run in Disk Basic, execution would continue to line 20. ■

```
10 FORA=1TO2 STEP0
20 X=X+1
30 B=B+1
40 C=C+1
50 IFX=100 PRINT"X=100":X=0:NEXTX
60 IFB=250 PRINT"B=250":B=0:NEXTX
70 IFC=3 PRINT"C=3":NEXTX
80 PRINTA
90 NEXTX
```

Program Listing 3.

```
10 CLS:PRINT@320,;
20 IFPEEK(16453)=0THENINPUT"ENTER DATE (MM/DD/YY) ";DT$:POKE1645
4,VAL(LEFT$(DT$,2)):POKE16453,VAL(MID$(DT$,4,2)):POKE16452,VAL(R
IGHT$(DT$,2))
30 PRINTLEFT$(TIME$,8);" IS DATE OK (Y/N)?"
40 B$=INKEY$:IFB$="N"THENPOKE16453,0:GOTO20:ELEIFB$="Y"THEN50ELS
E40
50 'PROGRAM STARTS, CLEAR, DIM, AND SO ON
60 '----- PROGRAM TEXT -----
70 '----- PROGRAM TEXT -----
80 '----- PROGRAM TEXT -----
```

Program Listing 4.

Know all the angles before you decide.

Cadet—The Decision Maker

Victor T. Albino
18501 194 NE
Woodinville, WA 98072

Cadet is designed to help you make logical decisions about almost any kind of problem. It provides an orderly process so each factor that goes into your decision making is listed and considered.

Cadet will provide all necessary prompts and protect against illegal entries so that you can easily state your problem and enter any factors you think important. Cadet will then make a recommendation for or against the action you have in mind.

The program is structured on the classic decision grid concept. This management tool facilitates decision making when a problem has many conflicting elements. The real value of Cadet is that it allows you to concentrate on identifying the issues and your feelings about them. The program takes care of the mathematics and even calculates a level of confidence for the suggestion it makes so you have some idea of how sure you can be about its advice.

The program also provides some cheerful graphics to keep your spirits high while your mind contemplates the issues. When you run the program, for example, you will see an animated spiral that simulates the "thinking" process of a RAM chip.

The action comes from lines 907 to 909. For some interesting variations on this theme, type in the following short program:

```
0 CLS
10 R=3
20 FOR X=0 to 375 step 7
30 B=R*7*SIN(X)+71
40 C=R*3*COS(X)+20
50 SET(B,C)
60 R=R-.05
70 NEXT X
80 GOTO 80
```

This is the same spiral pattern that appears in the program. Now try changing the 375 in line 20 to 850 and decrease the step size to 5: you get a pretty flower spiral.

Lines 30 and 40 plot the sine and cosine values of X using polar coordinates. The factors of 7 and 3 in these same lines are needed to compensate because the pixel (picture element) on your screen is not square; it is a rectangle with an end to side ratio of 3:7. If you start experimenting with these numbers, you will get an eliptical spiral instead of the circular one.

The numbers added at the ends of lines 30 and 40 are there to position the design on the CRT. It is best to keep the center of the design near the middle of your screen since an illegal function error results when the equation attempts to plot points off the video.

There are many modifications you can make in this program. By changing the values, often only slightly, you can create some fascinating graphics. After all, now that Cadet has solved your problems, you should have plenty of time to create fantastic graphics. ■

Program Listing

```
 ' C A D E T:  COMPUTER ASSISTED DECISION EVALUATION TECHNIQUE
   VER.3.0              BY VICTOR T. ALBINO
                                                 AUGUST  1980
4 CLS:CLEAR 3000:GOSUB 575:GOSUB748
5 CLS
15 PRINT:PRINT:PRINT:PRINT:PRINT"iT WILL DO THIS BY HELPING YOU
TO THINK THROUGH THE
VARIOUS POSITVE AND NEGATIVE FACTORS WHICH MUST BE CONSIDERED
BEFORE YOU CAN ARRIVE AT AN APPROPRIATE SOLUTION.":PRINT
25 PRINTTAB(40)"PRESS / KEY."
26 IF INKEY$<>"/"THEN26ELSECLS
30 PRINT:PRINT"IF YOU ARE READY, WE WILL BEGIN.":PRINT:PRINT"FIR
ST, YOU MUST TELL ME YOUR CONTEMPLATED ACTION.  IT SHOULD BE ENT
ERED AS BRIEFLY AS POSSIBLE AND PUT IN THE FORM OF A QUESTIONTHA
T CAN BE ANSWERED 'YES' OR 'NO'."
35 PRINT:PRINT"FOR EXAMPLE, A TYPICAL QUESTION MIGHT BE:
     'SHOULD I BUY THE USED CAR?'":PRINT
39 DIM A$(20)
40 PRINT"TYPE IN YOUR QUESTION NOW AND THEN PRESS <ENTER>."
45 INPUT ZZ$:CLS
50 PRINT:PRINTTAB(23)"THANK YOU."
55 PRINT:PRINT"NEXT YOU WILL BE ASKED TO ENTER STATEMENTS ABOUT
THE ACTION     YOU ARE CONTEMPLATING.  YOU MAY GIVE ME UP TO 20
STATEMENTS     TO CONSIDER."
60 PRINT:PRINT"BE SURE TO INCLUDE ALL THE FACTORS YOU CAN THINK
OF ON BOTH     SIDES OF THE ISSUE, THOSE IN SUPPORT AND THOSE AG
AINST."
65 PRINT:PRINT"IF THE STATEMENT IS IN FAVOR OF THE ACTION, PRECE
DE
YOUR STATEMENT WITH AN 'F' FOR 'FAVOR'."
70 PRINT"IF THE STATEMENT IS AGAINST THE ACTION, PRECEDE YOUR
STATEMENT WITH AN 'A' FOR 'AGAINST'.";:PRINT:PRINTTAB(40)"PRESS
/ KEY.";
75 IFINKEY$<>"/"THEN75ELSECLS
80 PRINT"SINCE I HAVE NO VALUE SYSTEM, I MUST RELY ON YOU."
85 PRINT:PRINT"AFTER EACH STATEMENT YOU GIVE ME, PLEASE ENTER IN
 YOUR
EVALUATION OF ITS IMPORTANCE ON A SCALE OF 1 TO 9."
90 REM
95 PRINT"FOR EXAMPLE, A  '1' WOULD INDICATE THAT THE FACTOR IS N
OT
TOO SIGNIFICANT.":PRINT"A '9' WOULD BE USED FOR THOSE ITEMS YOU
CONSIDER REALLY
```

Program continues

Program continued

```
IMPORTANT."
96 PRINT"(STATEMENTS NOT PRECEDED BY AN 'F' OR 'A' OR ENDING WIT
H A VALUEFROM 1 TO 9 WILL BE VOIDED.  THEY SHOULD BE RE-ENTERED.
)";:PRINT
100 PRINT:PRINT"SOME EXAMPLES OF STATEMENTS WOULD BE:
          F   THE USED CAR HAS VERY LOW MILEAGE  7
          A   THE CAR IS OVER-PRICED BY $500  8
          A   THE TIRES NEED REPLACING  5
          F   THE CAR'S COLOR IS MY FAVORITE  2";
105 PRINTTAB(51)"PRESS / KEY.";
106 IFINKEY$<>"/"THEN106 ELSECLS
110 PRINT:PRINT"I WILL KEEP AN ACCOUNT OF ALL THE FACTORS YOU TH
INK
SHOULD BE CONSIDERED.  WHEN YOU ARE THROUGH, SIMPLY TYPE
'N' IN RESPONSE TO THE QUESTION :  'ANY MORE FACTORS I
SHOULD CONSIDER?'":PRINT
115 PRINT"I WILL THEN REVIEW WITH YOU ALL THE FACTORS YOU HAVE I
DENTIFIED."
120 PRINT"AFTERWARDS, I WILL GIVE YOU MY SUGGESTED DECISION WHIC
H
WILL BE LOGICALLY DERIVED FROM YOUR OWN VALUES."
125 PRINT"FINALLY, I WILL GIVE YOU A CONFIDENCE LEVEL FOR THE
DECISION I SUGGEST.":PRINT
130 PRINTTAB(40)"PRESS / KEY."
131 IFINKEY$<>"/"THEN131ELSECLS
135 PRINT"THE QUESTION IS: ";ZZ$
140 PRINT:PRINT"PLEASE BEGIN NOW TO ENTER THE STATEMENTS FOR AND
AGAINST
THE PROPOSED ACTION.  (DON'T FORGET TO PUT AN 'F' OR 'A' BEFORE
EVERY STATEMENT AND A NUMBER FROM 1 TO 9 AFTER EACH ONE.)"
141 PRINT:PRINT"  F  -OR-  A                     STATEMENT
  1  TO  9"
142 PRINTSTRING$(64,"-"):NO=1:MU=0:NU=0
145 INPUTA$(1):GOSUB246:GOSUB250
150 INPUTA$(2):GOSUB246:GOSUB250
155 INPUTA$(3):GOSUB246:GOSUB250
160 INPUTA$(4):GOSUB246:GOSUB250
165 INPUTA$(5):GOSUB246:GOSUB250
170 INPUTA$(6):GOSUB246:GOSUB250
175 INPUTA$(7):GOSUB246:GOSUB250
180 INPUTA$(8):GOSUB246:GOSUB250
185 INPUTA$(9):GOSUB246:GOSUB250
190 INPUTA$(10):GOSUB246:GOSUB250
195 INPUTA$(11):GOSUB246:GOSUB250
200 INPUTA$(12):GOSUB246:GOSUB250
205 INPUTA$(13):GOSUB246:GOSUB250
210 INPUTA$(14):GOSUB246:GOSUB250
215 INPUTA$(15):GOSUB246:GOSUB250
220 INPUTA$(16):GOSUB246:GOSUB250
225 INPUTA$(17):GOSUB246:GOSUB250
230 INPUTA$(18):GOSUB246:GOSUB250
235 INPUTA$(19):GOSUB246:GOSUB250
240 INPUTA$(20):GOSUB246:GOSUB250
245 PRINT:PRINT"THAT MAKES 20 STATEMENTS WHICH IS ALL I CAN HAND
LE
AT ONE TIME.  LET ME PROCESS THESE AND SEE WHAT KIND OF
DECISION I REACH.":FOR T=1TO4000:NEXT:CLS:GOTO254

246 MU=MU+1:NU=NU+1:FORX=MU TO NU:IF LEFT$(A$(X),1)="F" OR LEFT$
(A$(X),1)="A" THEN 247 ELSEPRINT"RE-ENTER YOUR LAST STATEMENT AN
D PRECEDE IT WITH 'F' OR 'A'.";:PRINT:A$(X)="A VOID STATEMENT 0"
:RETURN
247 NR=VAL(RIGHT$(A$(X),1)):IF ABS(NR)>0 AND ABS(NR)<10 THEN RET
URN ELSE PRINT"RE-ENTER THE LAST STATEMENT AND END IT WITH A VAL
UE FROM 1 - 9.";:PRINT:A$(X)="A VOID STATEMENT 0":RETURN
250 INPUT"ANY MORE QUESTIONS TO CONSIDER (Y/N)";M$
251 IFM$="Y"ORM$="N"THEN253
252 PRINT"PLEASE ANSWER ONLY 'Y' FOR YES OR 'N' FOR NO.":GOTO250

253 IFM$="N"THEN254ELSENO=NO+1:RETURN
254 FOR X=1TO20
255 IF LEFT$(A$(X),1)="F"THEN SF=SF+VAL(RIGHT$(A$(X),1))ELSE SA=
SA+VAL(RIGHT$(A$(X),1))
260 NEXT X
355 GOSUB765
360 PRINT@460,"YOU GAVE ME ";NO;" FACTORS TO CONSIDER.":PRINT@58
8,"THEY WERE :";:FORT=1TO2000:NEXT:CLS:NU=1
361 FORX=1TONO:FORS=15552TO15615STEP2:POKES,176:NEXT
362 FORS=16064TO16127STEP2:POKES,176:NEXT
365 PRINT@448,NU;".)   ";MID$(A$(X),2,LEN(A$(X))-2):NU=NU+1
370 FORT=1TO2000:NEXT:CLS:NEXTX
465 PRINT@465,"PRESS / KEY TO CONTINUE."
466 IFINKEY$<>"/"THEN466ELSECLS
470 PRINT:PRINT"AFTER CAREFULLY ANALYZING ALL THE INFORMATION YO
U HAVE
PROVIDED ON THE QUESTION: ":PRINT
475 PRINTZZ$
480 PRINT:IF SF>SA THENPRINT"I RECOMMEND IN FAVOR OF THE PROPOSE
D ACTION.":GOTO 488
485 PRINT:IF SA>SF THENPRINT"I RECOMMEND AGAINST THE PROPOSED AC
TION.":GOTO 488
486 IF SF=SA THENPRINT"I AM UNABLE TO SUGGEST AN ANSWER.
THE POSITIVE AND NEGATIVE ELEMENTS IN THIS ISSUE BALANCE
OUT EXACTLY."
487 PRINT:PRINT"SORRY I CAN'T HELP YOU.  I AM AFRAID YOU ARE GOI
NG
TO HAVE TO USE HUMAN INTUITION ON THIS ONE!":PRINT:PRINT"BETTER
LUCK NEXT TIME.";:PRINTTAB(40)"PRESS / KEY.";:GOTO540
488 PRINT:PRINT:PRINTTAB(40)"PRESS / KEY."
489 IF INKEY$<>"/"THEN489ELSECLS
490 IF SF>SA THEN CF=INT(100-(SA/SF)*100)
495 IF SA>SF THEN CA=INT(100-(SF/SA)*100)
500 PRINT@448,"MY RECOMMENDATION IS MADE WITH A CONFIDENCE LEVEL
 OF:";
505 IF SF>SA THEN PRINT CF;"%":FA=CF
510 IF SA>SF THEN PRINT CA;"%":FA=CA
511 FORT=1TO2000:NEXT:CLS:PRINT:PRINT:PRINT:PRINT:PRINT
515 IF FA<50 THENPRINT"UNFORTUNATELY, THIS IS NOT A VERY HIGH LE
VEL
OF CONFIDENCE.  YOU MIGHT JUST AS WELL FLIP A COIN!":GOTO526
520 IF FA<80 THENPRINT"THIS IS A MODERATE LEVEL OF CONFIDENCE.

YOU CAN FEEL REASONABLY COMFORTABLE WITH MY RECOMMENDATION.":GOT
O526
525 IF FA>80 THENPRINT"THIS IS A HIGH LEVEL OF CONFIDENCE.
```

Program continues

Program continued

```
NOTHING IS SURE IN LIFE EXCEPT DEATH AND TAXES BUT THIS
EVIDENCE LOOKS PRETTY CONCLUSIVE."
526 PRINT:PRINT:PRINTTAB(40)"PRESS / KEY.";
527 IF INKEY$<>"/"THEN527
530 CLS:PRINT:PRINT:PRINT:PRINT:PRINT"NOW THAT YOU HAVE MY OPINI
ON YOU ARE, OF COURSE,
FREE TO DO AS YOU PLEASE.":PRINT
535 PRINT"HOWEVER, IF YOU ACCEPT MY ADVICE, YOU WILL KNOW THAT
YOUR DECISION WAS BASED UPON A COMPLETE EXAMINATION AND
COMPARISON OF ALL THE ISSUES.";:GOTO550
540 IFINKEY$<>"/"THEN540ELSE552
550 PRINT:PRINTTAB(40)"PRESS / KEY."
551 IF INKEY$<>"/"THEN551 ELSECLS
552 GOSUB736:POKE16025,188:FORS=16089TO16345STEP64:POKES,191:NEX
T:POKE16216,138:S=16342:POKES,176:POKES+1,184:POKES+2,188:S=1603
2:POKES,188:POKES+64,191:POKES+128,191:POKE16223,130:POKE16224,1
31:POKE16225,140
553 POKE16226,176:POKE16291,131:POKE16292,188:POKE16293,176:POKE
16294,144:Y=45:FORX=71TO76:SET(X,Y):NEXT:Y=46:FORX=69TO73:SET(X,
Y):NEXT
554 LU$=CHR$(160)+CHR$(160)+CHR$(168)+CHR$(168)+CHR$(160):LL$=CH
R$(130)+CHR$(190)+CHR$(190)+CHR$(190)+CHR$(190)
555 RU$=CHR$(188)+CHR$(190)+CHR$(188)+CHR$(188):RL$=CHR$(133)+CH
R$(149)+CHR$(149)+CHR$(133)+CHR$(133)
556 S=15751:POKES,160:POKES+1,134:FORX=S+2TOS+7:POKEX,131:NEXT:F
ORX=S+41TOS+46:POKEX,131:NEXT:POKES+47,129
557 POKE15814,152:POKE15815,129:POKE15734,160:POKE15735,134:POKE
15672,152:POKE15673,129:POKE15609,149:POKE15545,149:POKE15877,17
0:POKE15941,170
558 PRINT@54,LU$;:PRINT@118,LL$;:PRINT@644,RU$;:PRINT@708,RL$;
559 Y=15:FORX=29TO11STEP-2:SET(X,Y):NEXT:SET(11,14):Y=13:FORX=11
TO22STEP2:SET(X,Y):NEXT:X=22:FORY=13TO7STEP-2:SET(X,Y):NEXT
560 Y=7:FORX=21TO1STEP-2:SET(X,Y):NEXT:X=1:FORY=7TO13STEP2:SET(X
,Y):NEXT:Y=13:FORX=1TO11STEP2:SET(X,Y):NEXT
561 PRINT@194,"GOOD-BYE";
562 FORWA=1TO8:PRINT@54,CHR$(30);:FORT=0TO400:NEXT:PRINT@54,LU$;
:FORT=0TO500:NEXT:NEXT:CLS
563 END
575 N=15751'*OPENING GRAPHIC
576 BO=15360:PRINT@1,STRING$(62,137):PRINT@961,STRING$(62,164);
577 FORX=0TO1023STEP64:POKEBO+X,166:POKEBO+X+1,166:POKEBO+X+62,1
66:POKEBO+X+63,166:NEXT
580 POKEN,191:POKEN+1,191:FORX=N+2TON+5:POKEX,143:NEXT
585 N=N+64
590 POKEN,191:POKEN+1,191
595 N=N+64
600 POKEN,191:POKEN+1,191:FORX=N+2TON+5:POKEX,188:NEXT
605 O=15762:POKEO,191:POKEO+1,191:POKEO+2,143:POKEO+3,143:POKEO+
4,191:POKEO+5,191
610 O=O+64
615 POKEO,191:POKEO+1,191:POKEO+2,188:POKEO+3,188:POKEO+4,191:PO
KEO+5,191
620 O=O+64
625 POKEO,191:POKEO+1,191:POKEO+4,191:POKEO+5,191
630 P=15773
635 POKEP,191:POKEP+1,191:POKEP+2,143:POKEP+3,143:POKEP+4,191:PO
KEP+5,188
640 P=P+64
645 POKEP,191:POKEP+1,191:POKEP+4,191:POKEP+5,191
650 P=P+64
655 POKEP,191:POKEP+1,191:POKEP+2,188:POKEP+3,188:POKEP+4,191:PO
KEP+5,143
660 Q=15783
665 POKEQ,191:POKEQ+1,191:FORX=Q+2TOQ+5:POKEX,143:NEXT
670 Q=Q+64
675 POKEQ,191:POKEQ+1,191:POKEQ+2,143:POKEQ+3,143
680 Q=Q+64
685 POKEQ,191:POKEQ+1,191:FORX=Q+2TOQ+5:POKEX,188:NEXT
690 R=15794
695 POKER,143:POKER+1,143:POKER+2,191:POKER+3,191:POKER+4,143:PO
KER+5,143
700 R=R+64
705 POKER+2,191:POKER+3,191
710 R=R+64
715 POKER+2,191:POKER+3,191
720 FORT=1TO1000:NEXT
725 W$="    COMPUTER    ASSISTED    DECISION    EVALUATION    TECHNIQ
UE"
730 CLS:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
735 FORX=1TO59:PRINTMID$(W$,X,1);:FORT=1TO15:NEXTT,X:FORT=1TO100
0:NEXT
736 REM * COMPUTER GRAPHIC
737 CLS:S=15375:POKES,191:FORX=S+1TOS+3:POKEX,131:NEXT:FORX=S+4T
OS+23:POKEX,179:NEXT:FORX=S+24TOS+31:POKEX,131:NEXT:POKES+32,191

738 S=15439:POKES,191:POKES+2,188:POKES+3,131:POKES+24,131:POKES
+25,188:POKES+29,131:POKES+32,191
739 S=15503:POKES,191:POKES+2,191:POKES+8,131:POKES+9,135:POKES+
10,129:POKES+13,160:POKES+14,144:POKES+18,131:POKES+19,135:POKES
+20,129:POKES+25,191:POKES+28,191:POKES+29,191:POKES+30,191:POKE
S+32,191
740 S=15567:POKES,191:POKES+2,191:POKES+8,160:POKES+13,142:POKES
+14,141:POKES+19,144:POKES+25,191:POKES+28,131:POKES+29,131:POKE
S+30,131:POKES+32,191
741 S=15631:POKES,191:POKES+2,191:POKES+9,137:FORX=S+10TOS+17:PO
KEX,176:NEXT:POKES+18,134:POKES+25,191:POKES+32,191
742 S=15695:POKES,191:POKES+3,131:FORX=S+4TOS+23:POKEX,140:NEXT:
POKES+24,131:POKES+32,191
743 S=15759:POKES,191:FORX=S+1TOS+5:POKEX,176:NEXT:FORX=S+6TOS+2
1:POKEX,186:NEXT:FORX=S+22TOS+31:POKEX,176:NEXT:POKES+32,191
744 S=15821:FORX=STOS+36:POKEX,176:NEXT
745 S=15885:POKES,191:FORX=S+4TOS+24STEP2:POKEX,140:NEXT:FORX=S+
29TOS+33STEP2:POKEX,140:NEXT:POKES+36,191
746 S=15949:POKES,191:FORX=S+4TOS+24STEP2:POKEX,179:NEXT:FORX=S+
29TOS+33STEP2:POKEX,179:NEXT:POKES+36,191
747 S=16013:POKES,143:FORX=S+1TOS+35:POKEX,140:NEXT:POKES+36,143
:FORT=1TO400:NEXT:RETURN
748 Y=16:FORX=29TO10STEP-2:SET(X,Y):NEXT:X=10:FORY=17TO34STEP2:S
ET(X,Y):NEXT:Y=35:FORX=0TO127STEP2:SET(X,Y):NEXT:X=127:FORY=35TO
47STEP2:SET(X,Y):NEXT:Y=47:FORX=127TO0STEP-2:SET(X,Y):NEXT:X=0:F
ORY=47TO35STEP-2:SET(X,Y):NEXT
749 PRINT@770,"HELLO!  I'M YOUR COMPUTER.  THE PROGRAM YOU HAVE
PUT INTO MY";
750 PRINT@834,"MEMORY WILL ASSIST YOU TO REACH A LOGICAL DECISIO
N ABOUT ANY";
```

Program continues

Program continued

```
751 PRINT@898,"PROPOSED ACTION YOU MAY BE CONTEMPLATING.   (PRES
S '/ ' KEY.)";
752 IF INKEY$<>"/"THEN752
753 CLS:RETURN
765 REM * HEAD
770 CLS:S=15513
775 POKES,160:POKES+1,176:POKES+2,176:POKES+3,152
780 FORX=S+4TOS+14:POKEX,140:NEXT
785 FORX=S+15TOS+17:POKEX,176:NEXT
790 S=15574
795 POKES,160:POKES+1,152:POKES+2,134:POKES+3,129
800 POKES+20,130:POKES+21,137:POKES+22,164:POKES+23,144
805 S=15637
810 POKES,152:POKES+1,129:POKES+24,130:POKES+25,164
815 S=15700
820 POKES,152:POKES+1,129:POKES+27,137:POKES+28,144
825 S=15764
830 POKES,149:POKES+3,143:POKES+4,132:POKES+28,170
835 S=15826
840 POKES,152:POKES+1,134:POKES+30,170
845 S=15889
850 POKES,182:POKES+1,176:POKES+2,176:POKES+30,160:POKES+31,133
855 S=15956
860 POKES,169:POKES+5,144:POKES+25,152:POKES+26,134:POKES+27,129

865 S=16021
870 POKES,167:POKES+1,147:POKES+2,131:POKES+3,131:POKES+4,129:PO
KES+20,176:POKES+21,140:POKES+22,134:POKES+23,131
875 S=16086
880 POKES,130:POKES+1,131:POKES+2,140:POKES+3,176:POKES+4,176:PO
KES+16,152:POKES+17,134:POKES+18,131
885 S=16155
890 POKES,149:POKES+10,170
895 S=16219
900 POKES,181:FORX=S+1TOS+9:POKEX,176:NEXT:POKES+10,186:PRINT@26
,"WORKING RAM CHIP";:FORT=1TO500:NEXT
905 FORH=1TO6:PRINT@347,"T H I N K I N G";:FORT=1TO250:NEXT:PRIN
T@347,"                 ";:FORT=1TO100:NEXT:NEXT:FORT=1TO500:NEXT

906 REM * WHEELS TURNING
907 PRINT@245,"WHEELS";:PRINT@312,"TURNING!";:PRINT@371,"<";:PRI
NT@372,STRING$(9,"=");:R=3:FORT=1TO800:NEXT:FORX=0TO375STEP7
908 B=R*7*SIN(X)+73:C=R*3*COS(X)+20
909 SET(B,C):R=R-.05:NEXTX
910 FORT=0TO700:NEXT:S=15936
915 POKES,188:FORX=S+1TOS+14:POKEX,140:NEXT:POKES+15,188
920 S=16000
925 POKES,191:POKES+15,191:POKES+16,176:POKES+17,140:POKES+18,13
1
930 S=16064
935 POKES,143:FORX=S+1TOS+14:POKEX,140:NEXT:POKES+15,143
940 PRINT@642,"OK I'M READY";:FORT=1TO1000:NEXT:CLS:RETURN
```

Let the good times roll.

Micro-Yahtzee

Mark E. Johnson
2925 W. Lincoln Ave. #11
Anaheim, CA 92801

Here's a computer dice game to help you while away your time. I call it Micro Yahtzee.

The game first displays the instructions. Play begins when your roll is sorted into order and displayed. Use the INKEY$ function for all the entries so they are quick and easy. During the game the scores for each category appear continuously and the display indicates whether each category has been used or voided yet. After you roll, select your category. The program checks your dice against your selection and then tells you to make another selection (if you have used that category already), or gives you zero for that category if the dice don't apply, or gives you the appropriate points. At the end of the game your point totals and bonuses are displayed. Then the averages for each category and for each player for the games played are figured and displayed.

You will enjoy this computer version of Yahtzee because of the easy to read display and the safeguards against errors. ■

Lines 0–105	a. Initialize variables
	b. Display categories and scores
Lines 106–3707	a. Roll dice and change selected dice
	b. Category selection
Lines 3800–3960	subroutine puts new scores into active players' variables
Lines 4000–4030	subroutine counts dice for bonus selection
Lines 5000–5100	subroutine checks for 3 or 4 of a kind
Lines 6000–6190	subroutine checks for full house
Lines 7000–7200	subroutine checks for small or large straight
Lines 8000–8010	subroutine checks for Yahtzee
Lines 8500–8520	subroutine totals dice for chance
Lines 10000–10200	subroutine sets up temporary set of category numbers
Lines 11000–11075	a. Check for bonus eligibility
	b. Totals scores for game
	c. Check for new high or low score
Lines 12000–12100	subroutine sets player one's scores to strings
Lines 12200–12225	subroutine sets player two's scores to strings
Line 15000	subroutine clears portion of screen
Line 15005	subroutine clears portion of screen
Lines 16000–16060	subroutine sorts dice into order
Lines 17000–18000	totals all scores & displays summary
Lines 19000–19090	instructions for game

Table 1. Line Description

Program Listing

```
0 CLEAR150:DIM M(72):M(49)=1:M(50)=2:M(51)=3:M(52)=4:M(53)=5:M(5
4)=6:M(65)=7:M(66)=8:M(67)=9:M(68)=10:M(69)=11:M(70)=12:M(71)=13
:M(72)=14:LT=500
1 CLS:PRINT" M I C R O - Y A H T Z E E":PRINT:PRINT"INSTRUCTIONS
? (Y) OR (N)";
2 N$=INKEY$:IFN$=""THEN2
3 IFN$="Y"THEN19000
5 RANDOM:C1=1:DIMP1(14),B(13),B$(13),P2(14),A(14),P1$(13),P2$(13
),P3(14),P4(14)
6 CLS:INPUT"ENTER 1ST PLAYER'S NAME";P1$:INPUT"ENTER 2ND PLAYER'
S NAME";P2$
10 G1=0:CLS:FORA=1TO13:A(A)=.5:P1(A)=.5:P1$(A)=" *":P2$(A)=" *":
P2(A)=.5:NEXTA:A(14)=.5:CLS
62 PRINT@896,"
              ":GOSUB10000
65 FORA=1TO13:READB(A):NEXTA:FORA=1TO13:READB$:PRINT@B(A),B$;P1$
(A);"  ";:PRINT@B(A)+23,P2$(A):NEXTA:IFG1=26THEN11000
66 PRINT@768,P3$;"'S TURN";
67 ONA1GOSUB15000,15005
80 DATA35,99,163,227,291,355,419,483,547,611,675,739,803
90 DATA1-ONES------------,2-TWOS------------,3-THREES----------,
4-FOURS-----------,5-FIVES-----------,6-SIXES-----------,A--3 OF
 A KIND----,B--4 OF A KIND----,C--FULL HOUSE-----, D--SM. STRAIG
HT---, E--LG. STRAIGHT---, F--YAHTZEE--------, G--CHANCE--------
-
105 RESTORE
106 PRINT@0,"Y A H T Z E E";
3000 Q=911:S=0
3200 FORX=1TO5:H(X)=INT(RND(0)*6)+1:NEXTX
3225 GOSUB16000
3300 PRINT@896,"YOUR ROLL IS..";H(1);H(2);H(3);H(4);H(5);
3400 PRINT@925,"   HOW MANY CHANGES PAL  ";:N$=INKEY$:IFN$=""TH
EN3400ELSEN=VAL(N$)
3405 IFN$<"0"ORN$>"5"THEN3400
3500 PRINT@925,"                WHICH ";"(";N;")";" ";
3505 IFN=5THENP(1)=1:P(2)=2:P(3)=3:P(4)=4:P(5)=5:GOTO3580
3525 IFN=0S=2:GOTO3590
3530 FORO=1TON
3535 P$(O)=INKEY$:IFP$(O)=""THEN3535
3540 PRINT@953-N,"  ";:IFP$(O)<"0"ORP$(O)>"5"THEN3535ELSEP(O)=VA
L(P$(O)):PRINT@1005,"CHANGED #";P(O);
3570 NEXTO
3575 FORN2=1TO50:NEXTN2:PRINT@1005,"             ";
3580 FORQ=1TON:H(P(Q))=INT(RND(0)*6)+1:NEXTQ
3590 C=0
3600 FORQ=1TO4
3610 IFH(Q)<=H(Q+1)THEN3650
3620 T=H(Q):H(Q)=H(Q+1):H(Q+1)=T:C=C+1
3650 NEXTQ
3660 IFC>0THEN3590
3670 PRINT@896,"YOUR ROLL IS..";H(1);H(2);H(3);H(4);H(5);:S=S+1
3690 IFS<2THEN3400
3700 PRINT@896,"FINAL ROLL--->";H(1);H(2);H(3);H(4);H(5);"
     ENTER OPTION     ";:N$=INKEY$:IFN$=""THEN3700
3701 T1=ASC(N$)
3702 IFT1<49ORT1>54ANDT1<65ORT1>72THEN3700
3703 T=M(T1)
```

Program continues

Program continued

```
3704 IFT<1ORT>14THEN3700
3705 IFA(T)<>.5THENPRINT@1000,"YOU USED THAT ALREADY";:GOTO3700
3706 PRINT@1000,"                    ---> 0";N$;
3707 G1=G1+1
3800 ONTGOSUB4000,4000,4000,4000,4000,4000,5000,5000,6000,7000,7
000,8000,8500,11000
3900 IFC1/2<>INT(C1/2)THEN3950
3910 FORA=1TO13:P1(A)=A(A):NEXTA
3915 GOSUB12000
3920 GOTO62
3950 FORA=1TO13:P2(A)=A(A):NEXTA
3951 GOSUB12200
3960 GOTO62
4000 V=0
4005 FORU=1TO5
4010 IFH(U)=TTHENV=V+1
4020 NEXTU
4030 A(T)=V*T:RETURN
5000 IFT=7THENB5=3
5010 IFT=8THENB5=4
5020 FORB2=1TO5
5030 FORB3=1TO5
5040 IFH(B2)=H(B3)THENB6=B6+1
5050 NEXTB3
5060 IFB6<B5THENB6=0
5070 IFB6>=B5THEN5100
5080 NEXTB2
5085 B6=0:A(T)=0:RETURN
5100 A(T)=H(1)+H(2)+H(3)+H(4)+H(5):B6=0:RETURN
6000 M3=1
6020 FORF2=2TO5
6030 IFH(1)=H(F2)THEN6100
6040 M2=M2+1:D(M2)=H(F2):GOTO6110
6100 M3=M3+1
6110 NEXTF2
6120 FORE1=1TOM2
6130 IFD(1)=D(E1)THEN6150
6140 A(9)=0:M2=0:RETURN
6150 NEXTE1
6160 IFM2>3THEN6140
6170 IFM3>3THEN6140
6180 IF(M2+M3)<>5THEN6140
6190 A(9)=35:M2=0:RETURN
7000 G3=0:L1=0
7010 IFT=10THEN7100
7020 IFH(1)=H(2)-1ANDH(2)=H(3)-1ANDH(3)=H(4)-1ANDH(4)=H(5)-1THEN
7200
7030 A(11)=0:RETURN
7100 W1=0
7102 FORA=1TO5:W(A)=0:NEXTA
7103 FORA=1TO5
7104 IFH(A)=H(A+1)THEN7107
7105 W1=W1+1
7106 W(W1)=H(A)
7107 NEXTA
7108 IFW(1)+1=W(2)ANDW(2)+1=W(3)ANDW(3)+1=W(4)THEN7150
7109 IFW(2)+1=W(3)ANDW(3)+1=W(4)ANDW(4)+1=W(5)THEN7150
7122 A(10)=0:RETURN
7150 A(10)=35:RETURN
7200 A(11)=40:RETURN
8000 IFH(1)=H(2)ANDH(2)=H(3)ANDH(3)=H(4)ANDH(4)=H(5)THENA(12)=50
:RETURN
8010 A(12)=0:RETURN
8500 A(13)=0
8510 FORA=1TO5:A(13)=A(13)+H(A):NEXTA
8520 RETURN
10000 C1=C1+1:IFC1/2<>INT(C1/2)THEN10100
10005 P3$=P1$
10006 A1=1
10010 FORA=1TO13:A(A)=P1(A):NEXTA
10020 RETURN
10100 P3$=P2$
10101 A1=2
10200 FORA=1TO13:A(A)=P2(A):NEXTA:RETURN
11000 F7=0
11005 FORA=1TO6:F7=F7+P1(A):NEXTA
11008 IFF7<63THENF7=0
11010 IFF7>62THENF7=35
11012 IFF7=35THENPO=PO+1:PO(1)=35
11020 FORA=1TO13:B7=B7+P1(A):NEXTA
11030 B7=B7+F7:T1=INT(B7):B7=0
11040 F7=0:FORA=1TO6:F7=F7+P2(A):NEXTA
11043 IFF7<63THENF7=0
11045 IFF7>62THENF7=35
11046 IFF7=35THENPT=PT+1:PT(1)=35
11055 FORA=1TO13:B7=B7+P2(A):NEXTA
11060 B7=B7+F7:T2=INT(B7):PRINT@256,P1$;"=";T1,"BONUS=";PO(1);:P
O(1)=0:PRINT@320,P2$;"=";T2,"BONUS=";PT(1);:PT(1)=0:B7=0:P1(14)=
T1:P2(14)=T2
11065 IFT1>T2THENW4=W4+1
11066 IFT1<T2THENW2=W2+1
11067 IFT1=T2THENW3=W3+1
11070 T3=T3+T1:T4=T4+T2:T5=T5+1:PRINT@384,"PUSH ANY KEY TO CONTI
NUE";
11071 L5$=INKEY$:IFL5$=""THEN11071
11074 GOSUB17000
11075 GOTO10
12000 FORA=1TO13
12005 P1$(A)=STR$(P1(A))
12010 IFP1(A)=0THENP1$(A)=" X"
12015 IFP1(A)=.5THENP1$(A)=" *"
12020 NEXTA
12100 RETURN
12200 FORA=1TO13
12205 P2$(A)=STR$(P2(A))
12210 IFP2(A)=0THENP2$(A)=" X"
12215 IFP2(A)=.5THENP2$(A)=" *"
12220 NEXTA
12225 RETURN
15000 PRINT@886,"[";:RETURN
15005 PRINT@891,"[";:RETURN
16000 C=0
16010 FORQ=1TO4
16020 IFH(Q)<=H(Q+1)THEN16040
16030 T=H(Q):H(Q)=H(Q+1):H(Q+1)=T:C=C+1
```

Program continues

✓See List of Advertisers on page 386

Program continued

```
16040 NEXTQ
16050 IFC>0THEN16000
16060 RETURN
17000 CLS:FORA=1TO14:P3(A)=P3(A)+P1(A):P4(A)=P4(A)+P2(A):NEXTA
17005 READA$:IFA$="START"THEN17100ELSE17005
17100 PRINT"    ";P1$,P2$
17200 A8=82:FORA=1TO14:READA$:PRINTA$;"    ";:PRINTINT(P3(A)/T5+.
49);:PRINT@A8,INT(P4(A)/T5+.49):A8=A8+64:NEXTA
17300 IFHT<=P1(14)THENHT=P1(14):HT$=P1$
17400 IFLT>=P1(14)THENLT=P1(14):LT$=P1$
17500 IFHT<=P2(14)THENHT=P2(14):HT$=P2$
17600 IFLT>=P2(14)THENLT=P2(14):LT$=P2$
17700 PRINT@92,"LOWEST SCORE WAS BY ";LT$;"=";LT;:PRINT@156,"HIG
HEST SCORE WAS BY ";HT$;"=";HT;:PRINT@220,P1$;"'S";" BONUS =";PO
;" OF ";T5;:PRINT@284,P2$;"'S";" BONUS =";PT;" OF ";T5;:PRINT@41
2,P1$;:PRINT@422,P2$;:PRINT@432,"TIES";
17705 PRINT@476,W4;:PRINT@486,W2;:PRINT@496,W3;:PRINT@960,"(AVE.
 PTS. PER GAME FOR EACH CATAGORY)";
17800 N$=INKEY$:IFN$=""THEN17800
17900 RESTORE:RETURN
18000 DATA START,1=,2=,3=,4=,5=,6=,A=,B=,C=,D=,E=,F=,G=,AV
19000 CLS:PRINT@21,"THE GAME INVOLVES THE 'ROLLING' OF FIVE    D
ICE AND GETTING COMBINATIONS THAT MATCH EACH OF THE 13 CATA-   G
ORIES. (E.G. 3 OF A KIND, LARGE STRAIGHT ETC.)
19002 PRINT@274,"ON EACH TURN YOU GET TO ROLL THE DICE 3 TIMES A
ND CAN DECIDE ON ROLLS TWO AND THREE TO RE-ROLL JUST THE ONES  Y
OU WANT. WHEN THE COMPUTER ASKS 'WHICH ONES' YOU RESPOND WITH  T
HE  DIE NUMBER (1 TO 5  LEFT TO RIGHT)."
19004 PRINT@594,"AFTER THE LAST ROLL YOU MUST CHOOSE A CATAGORYE
VEN IF YOUR ROLL DOES NOT MATCH ONE, YOU MUST CHOOSE ONE AND IFY
OUR ROLL DOESN'T MATCH THEN THAT CATAGORY WILL JUST BE 'X' OUT.T
HE 'BONUS' CATAGORIES ARE NUMBERS 1 TO 6."
19005 PRINT@813,"YOU GET A BONUS OF 35 POINTS IF YOUR TOTAL FOR
CATAGORIES 1 TO 6 IS 63 OR MORE.":PRINT:INPUT"PUSH ENTER TO CONT
INUE";X
19010 CLS:PRINT"FOLLOWING ARE THE POINTS FOR EACH CATAGORY"
19015 PRINT:PRINT"1 TO 6  BONUS POINTS. DIES MATCHING CATAGORY N
UMBER ARE ADDED"
19020 PRINT"(A)  THREE OF A KIND. ONLY WITH THREE OF A KIND, ALL
 DICE ADDED"
19030 PRINT"(B)  FOUR OF A KIND. ONLY WITH FOUR OF A KIND, ALL D
ICE ADDED"
19040 PRINT"(C)  FULL HOUSE. 3 AND 2 OF ANY KIND. 35 POINTS"
19050 PRINT"(D) SMALL STRAIGHT. 4 IN SEQUENCE. 35 POINTS"
19060 PRINT"(E) LARGE STRAIGHT. 5 IN SEQUENCE. 40 POINTS"
19070 PRINT"(F) YAHTZEE. ALL FIVE SAME. 50 POINTS"
19080 PRINT"(G) CHANCE. ANY COMBINATION. ALL DICE ADDED"
19082 PRINT"(H)  WILL END GAME IMMEDIATELY"
19084 PRINT"NOTE: PUSHING 5 AT 'HOW MANY CHANGES' RE-ROLLS ALL T
HE DICE":PRINT"NOTE: PUSHING 0 AT 'HOW MANY CHANGES' PROMPTS 'WH
ICH OPTION'"
19090 PRINT:INPUT"ENTER TO START GAME";X:GOTO1
```

✓See List of Advertisers on page 386

You sometimes must use them to get the most out of your disparate program library.

Building Bridges

Wayne L. Mueller
130 Sunset Court
Roswell, GA 30075

At one time or another, almost all of us have bought software that did everything that it claimed, but still didn't do exactly what we had in mind. When this happens, we usually either change our mind about what needs to be done, or change the software to do what we originally had in mind. In this case, I did neither. Rather, I found two pieces of software and built a bridge between them. Taken together, they do exactly what I want.

Special Delivery

On one side of the bridge we have Special Delivery by Software Concepts. This is a very nice mail-list processor that merges a letter text file with a name and address file to produce highly personalized form letters.

If your letter starts with "Dear Mr. <L:", the Mailrite program will substitute a last name for the <L, so that it prints as Dear Mr. Adams, Dear Mr. Baker, or whatever last names may appear in your name and address file. In all, there are eight fields in the name and address file that may be substituted anywhere in the body of your letter. Other "<" flags allow you to toggle underscores and boldface on and off, change margins dynamically, and provide other slick features too numerous to mention here. (See the July, 1980, *80 Microcomputing* for a review.) The problem is that six of the eight fields have to do with the name and address, leaving only two fields to store data about that person or company. For my purposes, two is not enough.

AIDS3

Enter AIDS3, by Meta Technologies. AIDS has no text insertion capability, but is a superior data manager. The user may define up to 20 fields of either character or numeric type. In addition, there is a more comprehensive search and select capability. There are no exaggerations in MTC's ads. If anything, the claims are understated. If all you need is a data manager and report generator, AIDS3 and its companion programs are top notch.

The Bridge

As you might expect, the file structures differ between AIDS and Special Delivery, so that a file written by one cannot be read directly by the other. That's where the bridge comes in. The conversion Program Listing moves any of AIDS' 20 fields to or from any of Special Delivery's eight fields, and does it in a format that is readable by the destination program.

What this means to me is that even after using six of the 20 AIDS fields for name and address information, I still have up to 14 fields to store data for insertion into form letters. Normally I do my data entry, editing and selection with AIDS, and then move the selected data to the Special Delivery format for form letter generation.

The conversion program operates very simply. The program asks for and accepts the names of the files involved, and which way to convert. The respective file formats are then displayed, (Fig. 1) and you select which field is to be moved where. As each selection is made, the arrows to the left of the screen (Fig. 2) depict the source and destination fields. In the example shown, an AIDS file is being converted to Special Delivery. AIDS fields A through F are being moved to the corresponding Special Delivery fields one through six, while AIDS fields M and L are being moved to Special Delivery fields six and eight respectively. Entering the slash (/) character ends the selection, and causes the conversion to take place.

All Special Delivery fields are stored as left-justified characters, and padded on the right with underline characters to fill out the field. AIDS' character fields are padded on the right with blanks, while numeric fields are right justified and padded on the left with blanks. The conversion process accomodates these conventions, so that the converted file appears identical to what would have been produced by the destination program. If the destination field is smaller than

```
 Member/lst:1        Member/dat:1
1 Name 25           A Name = 25           K Pledge $#6
2 Company 25        B Company = 25        L $ to Date#6
3 Address 25        C Address = 25        M Committee#18
4 City 18           D City = 18           N Officer?#1
5 State 2           E State = 2
6 Zip 5             F Zip = 5
7 Data 1 14         G Wife Name = 12
8 Data 2 14         H = Children-3
                    I Member Yrs-4
                    J Birthday-18
                  To or From Member/dat:1?
```

Fig. 1

```
          Member/lst:1         Member/dat:1
A = => 1 Name 25          A Name-25          K Pledge $-6
B = => 2 Company 25       B Company-25       L $to Date-6
C = => 3 Address 25       C Address-25       M Committee-18
D = => 4 City 18          D City-18          N Officer?-1
E = => 5 State 2          E State-2
F = => 6 Zip 5            F Zip-5
M = => 7 Data 1 14        G Wife Name-12
L = => 8 Data 2 14        H # Children-3
                          I Member yrs-4
                          J Birthday-18
                                <= =
                               Next?
```

Fig. 2

TEXAS COMPUTER SYSTEMS
Offers Lowest Prices on

FOR PERSONAL BUSINESS
OR ENTERTAINMENT

Color Computer
4K Level I $319
16K Level I $439
16K Extended Basic $489

With TCS Memory:
16K Level I $369
16K Extended Basic $449

Expansion Interfaces
Zero K Interface $254
16K Interface $359
32K Interface $469

With TCS Memory:
16K Interface $318
32K Interface $388

Epson Printers $Call

Letter quality matrix printer has full software control of 40, 80, 66 or 132 columns 80 cps bidirectional tractor feed, disposable printhead $300 less than nearest competitive printer Lists **$645** Call for our low price

MX-80 Tractor Feed
MX-80 FT Friction and Tractor

MX-100 Graphtrax, Friction and Tractor up to 15 " wide.

Graphtrax for MX-80, MX-80 FT, graphics option.

SOFTWARE

**EPSON MX-80
PRINTER DISKETTE**

A complete diskette with 6 complete files for use with your Epson MX 80 Printer ready to use

M/CMD Send all of the Epson Printer commands to the printer directly from DOS or as part of an auto booting system Set up the printer from DOS to the typestyle you want then run the program

BIGLETT/BAS Produce large graphic letters on your printer and screen Dump the letters to printer or to disk Allows 3 different size letters to the printer Program is menu driven You may also construct large letter titles for MERGEing into your basic program

EPSON/SUB Merge this small module into any Basic program and then issue any of the printer commands with a 2 letter command with letters representing the purpose of the command You need not refer back and forth to documentation Extremely easy to use and makes using the printer much easier than ever before

JKL PATCH This patches NEWDOS 2 1 and NEWDOS 80 so that the JKL function will print the TRS80 graphics while the printer remains in the standard mode rather than the TRS80 mode This allows all of the features of the printer to be used but you can dump the screen to the printer at any time with all graphics

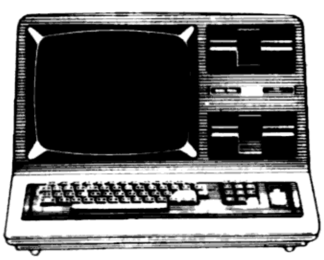
DEMO/BAS A super tutorial program with hundreds of REMs and hundreds of screen prompts that teach you how easy it is to create miracles with your Epson MX80 printer

LABEL/BAS – Custom label making program including graphics Makes custom mailing labels an easy job

DEFSTATE/SUB A series of one word printer commands that can be merged into any Basic Programs that allow various automatic functions to occur Print titles with graphic borders around them with one word Center any text with one command etc

Special Price All seven of the above modules on one diskette or on tape (less JKL) for your Model I or III for only **$29.95**

Or take a $10 discount if you buy this item with purchase of the Epson MX-80



TRS-80

We have all Radio Shack computers and accessories, and a number of other brand computer items Call us for your specific needs and get a copy of our catalog

Model III 16K $835
Model III 32K $979
Model III 48K $1089

With TCS Memory:
Model III 32K $909
Model III 48K $969

Model III 48K 2 Disk RS232 $2100
Model III 32K 1 Disk $1729
Model III 48K 1 Disk $1849

TEXAS COMPUTER SYSTEMS
Offers Lowest Prices on

TRS-80
COMPUTERS

Model II 64K $3288

An excellent computer for your business needs. Easy expandability & compatibility. No formal operator training needed. All accessories available—disk expansions, printers, software, at our low discount prices. Our fast, fully insured air freight service can assure most deliveries within seven days after payment is received.

Corvus Hard Disks $Call

5, 10, or 20 megabytes of storage for the Model I, II, or III. We can configure a system for you with either TRSDOS, NEWDOS 80, or CP/M for one, or several computers sharing a single hard drive simultaneously. For example, from 1 to 8 Model II's may share a single large data base with the Corvus Constellation Multiplexer. This also allows you to use different manufacturers computers on the same hard disk system at the same time. Also fits Apple, Superbrain, Altos and most other computers. As your business grows, additional storage and terminals may be added easily, inexpensively. Why wait months for other's promises? We can deliver these reliable proven hard disk systems NOW! Corvus has sold and shipped ten times more hard drives for microcomputers than all their competitors combined. **Call us for the lowest price!**

MICROSOFT

BASIC-80 Extended disk Basic, long variable names chaining variable length records ANSI compatable interpretative version **$294 w/manual. $30 manual only.**

BASIC COMPILER Same Basic language as BASIC 80 except compiles Basic program into a directly executable, relocatable machine language file Operates 3 to 10 times faster than regular Basic Includes Macro-80 Assembler Specify CPM or TRSDOS **$325 w/manual. $30 manual only.**

FORTRAN-80 Supports ANSI 66 plus many extensions including MACRO-80 Has library with manager, relocatable object compiler and linking loader **$400 w/manual. $30 manual only.**

Word Processor Package $2679

Includes 2 Disk Model III with 48K, Epson MX-80 Tractor Feed with cable, and word processing software ready to operate. Lists at $3300. Our low price special this month: **$2679.** For MX-80 FT Tractor and Friction, add $99.

Special—With TCS Memory
★ **Model III 48K 2 Disks** **$1895**

Uses proven MPI drives modified for flippy operation and Percom controller Limited 90 day warranty Call for details

Printers

Daisy Wheel II
$1695

Carbon ribbon typewriter quality for word processing 43 cps Standard Centronics parallel interface Full 16" width individual sheets or friction feed Tractor feed $250 extra Fastest Daisy Wheel available at this price

Line Printer IV $825
Line Printer V $1599
Line Printer VI $995
Line Printer VII $350

Pocket Computer & Acc. $Call

Pocket Computer Printer
Interface in stock

- Payment Money Order Cashier s Check Certified Check Personal checks take 3 wks VISA MC add 3%
- Prices subject to change any time
- No tax out-of-state Texans add 5%
- Delivery subject to availability
- Shipping extra quoted by phone ✓25

TEXAS COMPUTER SYSTEMS

Box 951, Brady Texas 76825

For fast, efficient service we can air freight from Dallas to major a/p near you. Call for information.

Toll Free Number 800-433-5184
Texas Residents 817-274-5625

the source field, the source is truncated on the right or left as appropriate to the destination.

No changes to Special Delivery and only minor changes to AIDS are required. Special Delivery handles lowercase characters while AIDS does not. Simply go through AIDS and locate the occurrences of CH = 90. Change them to CH = 122 and you have lowercase AIDS. This naturally assumes that your TRS-80 has a lowercase conversion installed. ■

```
100 DEFINT A-Z:CLEAR 1000
110 'DEFINE MACHINE CODE IN ML$
120 ML$=STRING$(26,32):X=VARPTR(ML$):X1=PEEK(X+1):X2=PEEK(X+2)
130 IF X2>127 THEN X2=-X2:X1=X1-1 ELSE X2=256*X2
140 X=X1+X2:DEFUSR0=X:FOR X1=0 TO 25:READ X2:POKE X+X1,X2:NEXT
150 CL$=CHR$(31) 'CLEAR SCREEN
160 GOSUB 3500 'GET FILE NAMES
170 GOSUB 2000 'OPEN & READ DESC FILE
180 GOSUB 2100 ' DISPLAY AIDS FIELDS
190 GOSUB 2200 'DISP SPEC DEL FIELDS
200 GOSUB 2300 'FIELD SD FILE
210 GOSUB 4700 'DUMMY AIDS FIELDING
220 GOSUB 2400 'CONVERT WHICH WAY?
230 GOSUB 2700 'OPEN AIDS FILE FOR I/O
240 LT$="ABCDEFGHIJKLMNOPQRST" 'VALID LETTERS
250 LT$=LEFT$(LT$,NF) 'VALID LETTERS...THIS AIDS SYSTEM
260 NR$="12345678" 'VALID NUMBERS FOR S-D
270 GOSUB 3800 'GET & VALIDATE XFER INFO
280 IF TF=1 THEN GOSUB 4000 ELSE GOSUB 4300
290 CLOSE:END
2000 'SUBR TO OPEN & READ AIDS DESC FILE
2010 OPEN"I",3,F3$
2020 INPUT#3,A$,A$,TL,NF 'DMY,DMY,TOT REC LEN,# FLDS
2030 DIM AN$(NF),AL(NF),AS(NF),AZ(NF),AA$(NF)'NAME,LEN,STRT,BKTS
2040 FOR I=1 TO NF:INPUT#3,AN$(I),AL(I) 'FLD NAME & LENGTH
2050 IF AL(I)<0 THEN INPUT #3,AS(I),AS(I) ELSE INPUT#3,AS(I)
2060 NEXT:CLOSE#3:RETURN
2100 'SUBR TO DISPLAY AIDS FIELDS
2110 CLS:PRINT@40,F2$; 'DISP AIDS NAME
2120 NS=686 'START PRINT POS
2130 FOR I=20 TO 1 STEP -1:IF I>NF THEN 2150
2140 PRINT@NS,CHR$(64+I);" ";AN$(I);"--";ABS(AL(I));
2150 NS=NS-64:IF NS<64 THEN NS=666
2160 NEXT:RETURN
2200 'SUBR TO DISP SPEC DEL FIELDS
2210 NS=5:PRINT@NS,F1$;'DISP S-D FILENAME
2220 FOR I=1 TO 8:NS=NS+64:READ SN$(I),SL(I)
2230 PRINT@NS,I;SN$(I);SL(I);:NEXT:RETURN
2300 'SUB TO FIELD RND FILE TO S-D SPECS
2310 OPEN"R",1,F1$:SR=2:SF=8 '2 SUBRECS, 8 FLDS
2320 DIM RF$(SR,SF),SZ(NF):DUMMY=0
2330 FOR I=1 TO SR:FOR J=1 TO SF
2340 FIELD 1,(DUMMY) AS DUMMY$,SL(J) AS RF$(I,J)
2350 DUMMY=DUMMY+SL(J):NEXT J,I:RETURN
2400 'CONVERT WHICH WAY?
2410 AR$(1)=" <==":AR$(2)=" ==>"
2420 NS=720:PRINT@NS,CL$;"TO OR FROM ";F2$;"?"
2430 A$=INKEY$:IF A$="" THEN 2430
2440 PRINT@NS,CL$;:TF=INSTR("TF",A$):IF TF>0 THEN 2460
2450 GOSUB 2600:GOTO 2420 'BUZZ & CLEAR
2460 SY$=AR$(TF):PRINT@NS+6,AR$(3-TF):RETURN
2600 'BEEP ROUTINE
2610 FOR I=1 TO 25:OUT 255,4:OUT 255,0:NEXT:RETURN
2700 'OPEN AIDS FILE FOR I/O
2710 OPEN MID$("OI",TF,1),2,F2$:RETURN
3500 ' SUB TO GET FILENAMES
3510 PRINT@0,CL$:PRINT@10,"MAILFORM/AIDS FILE CONVERSIONS"
3520 PRINT@268,"MAILFORM FILE? ";:LINEINPUT F$
3530 NS=283:F=1:GOSUB 3700:F1$=F$
3540 PRINT@336,"AIDS FILE? ";:LINEINPUT F$
3550 NS=347:F=2:GOSUB 3700:F2$=F$
3560 PRINT@389,"AIDS DESCRIPTOR FILE? ";:LINEINPUT F$
3570 NS=411:F=3:IF F$="" THEN F$=F0$:GOSUB 3750 ELSE GOSUB 3700
3580 F3$=F$:PRINT@471,"OK? ";
3590 LINEINPUTF$:IFLEFT$(F$,1)="Y" THEN RETURN ELSE GOTO 3510
3700 'ASSIGN DEFAULT EXT & DRIVE
3710 A1=INSTR(F$,"/"):A2=INSTR(F$,":"):FR$=":1"
3720 IF A1>0 AND A2>0 THEN RETURN
3730 IF A2>0 THEN FR$=RIGHT$(F$,2):F$=LEFT$(F$,A2-1)
3740 F0$=F$
3750 F$=F$+MID$("/LST/DAT/DSC",4*F-3,4)+FR$:PRINT@NS,F$:RETURN
3800 'SUB TO VALIDATE FLD XFER INFO
3810 NS=918:PRINT@NS,CL$;"NEXT? ";:NS=NS+6:FL=2:GOSUB 4500
3820 TX$=AI$:IF AI=42 THEN 3960 'STD XFER
3830 IF AI=47 THEN RETURN 'END INPUT
3840 L$=LEFT$(TX$,1):R$=RIGHT$(TX$,1)
3850 L=INSTR(NR$,L$):IF L>O THEN L1=1:GOTO 3870
3860 L=INSTR(LT$,L$):IF L>0 THEN L1=2:GOTO 3880 ELSE GOTO 3900
3870 R=INSTR(LT$,R$):IF R>0 THEN R1=2:GOTO 3910 ELSE GOTO 3900
3880 R=INSTR(NR$,R$):IF R>0 THEN R1=1:GOTO 3910
3890 'INVALID INPUT
3900 NS=852:PRINT@NS,TX$;" INVALID";:GOSUB 2610:GOTO 3810
3910 IF L1=R1 THEN 3900
3920 IFL1=1THEN X=R:Y=L ELSE IF L1=2 THEN X=L:Y=R ELSEGOTO3900
3930 AZ(X)=Y:SZ(Y)=X:NS=64*Y:PRINT@NS,CHR$(64+X)+SY$;
3940 PRINT@852,CL$;:GOTO 3810 'CLEAR ERR & RESUME
3950 'STD XFER
3960 FOR I=1 TO SF:SZ(I)=I:PRINT@64*I,CHR$(64+I)+SY$;:NEXT
3970 FOR I=1 TO NF:AZ(I)=I:NEXT
3980 GOTO 3940
4000 ' SUB TO CONV S-D TO AIDS
4010 B$=STRING$(TL,32):EC$=CHR$(236):UL$=CHR$(95):RN=0
4020 FOR K=1 TQ LOF(1):NR=NR+1:GET 1,NR
4030 FOR I=1 TO SR
4040 IF INSTR(RF$(I,1),EC$)>0 THEN 4130
4050 FOR J=1 TO SF 'FOR ALL FIELDS
4060 LSET AA$(J)=B$
4070 UL=INSTR(RF$(I,AZ(J)),UL$):IF UL=1 THEN 4110 ELSE IF UL=0 T
HEN 4090
4080 A$=LEFT$(RF$(I,AZ(J)),UL-1):GOTO 4100
4090 A$=RF$(I,AZ(J))
4100 IF AL(J)>0 THEN LSET AA$(J)=A$ ELSE RSET AA$(J)=A$
4110 PRINT#2,AA$(J);:NEXTJ:PRINT#2,""
4120 RN=RN+1:PRINT@714,RN;"RECORDS WRITTEN TO ";F2$:NEXT I,K
4130 CLOSE:END
4300 ' SUB TO CONV AIDS TO S-D
4310 NR=0:RC=0
4320 FOR I=1 TO SR 'FOR ALL SUB RECS
4330 IF EOF(2) THEN 4440
4340 LINEINPUT#2,A$:FOR J=1 TO SF 'FOR ALL FLDS
4350 T$=STRING$(LEN(RF$(I,J)),32)
4360 X$=MID$(A$,AS(SZ(J)),ABS(AL(SZ(J))))
4370 IF AL(SZ(J))>0 THEN 4410
4380 FOR K=1 TO LEN(X$):IF MID$(X$,K,1)<>" " THEN 4400
4390 NEXT K:GOTO 4410
4400 K=K-1:X$=RIGHT$(X$,LEN(X$)-K)
4410 MID$(T$,1)=X$
4420 X=USR0(VARPTR(T$)):LSET RF$(I,J)=T$:NEXT J
4430 RC=RC+1:PRINT@714,RC;"RECORDS WRITTEN TO ";F1$:NEXT I
4440 NR=NR+1:PUT 1,NR:IF EOF(2) THEN 4450 ELSE GOTO 4320
4450 CLOSE:END
4500 'KEYBOARD INPUT ROUTINE
4510 FS=0:AN$="":CR$=CHR$(13):CF$=CHR$(14):CN$=CHR$(15)
4520 PRINT@NS+FS,STRING$(FL-FS,136)
4530 PRINT@NS+FS,CF$;:I$=INKEY$:PRINT@NS+FS,CN$;:IFI$=AN$THEN453
0
4540 AI=ASC(I$)
4550 IF AI=42 OR AI=47 THEN PRINT@NS+FS,I$;:FS=FS+1:GOTO 4600
4560 IF AI=8 THEN IF FS>0 THEN FS=FS-1:GOTO 4530 ELSE 4530
4570 IF I$=CR$ THEN 4600
4580 PRINT@NS+FS,I$;:IF AI<>8 THEN FS=FS+1 ELSE FS=FS-1
4590 IF FS<FL THEN 4520
4600 AI$=AN$:BP=15360+NS:FOR I=BP TO BP+FS-1:AI$=AI$+CHR$(PEEK(I
)):NEXT:RETURN
4700 'DUMMY FIELDING FOR AIDS FILE
4710 OPEN"R",3,F3$:DUMMY=0:FOR I=1 TO NF
4720 FIELD 3,(DUMMY) AS DUMMY$,ABS(AL(I)) AS AA$(I)
4730 DUMMY=DUMMY+ABS(AL(I)):NEXT:RETURN
5000 DATA 205,127,10,06,01,05,78,35,94,35,86,235,9,43,65
5010 DATA 62,32,190,192,62,95,119,43,16,246,201
5020 DATA "NAME",25,"COMPANY",25,"ADDRESS",25,"CITY",18
5030 DATA "STATE",2,"ZIP",5,"DATA 1",14,"DATA 2",14
```

Program Listing

...And my dad says GRAFTRAX 80 does better graphics than anybody.

Epson.

You might say it's the head of its class. GRAFTRAX 80 gives you the highest resolution dot matrix in the world — up to 120 dots per inch horizontally and 216 dots per inch vertically. Which makes our MX-80 and MX-80 F/T work more like a plotter than a printer.

And that's just for starters.

GRAFTRAX 80 more than doubles the features available in an Epson MX-80 or MX-80 F/T. For instance, we've added nice little touches like an italic character set — which doubles the number of font types to 24. And with GRAFTRAX 80, you can change style, size and density anywhere in the line. And backspace the head. And use block graphics with *any* computer. And redefine all escape codes under software control.

Up to now, you couldn't get some of these features at *any* price. With GRAFTRAX 80, you get 'em all. And more. Lots more.

Best of all, we haven't forgotten all those people who already own an MX-80 or MX-80 F/T. You own the printer that set the standards all the others are following. And you can install GRAFTRAX 80 PROMs yourself, for less than $100. And make the best even better.

You can see why we say GRAFTRAX 80 is the head of its class. There's nothing else *in* its class.

404

EPSON

EPSON AMERICA, INC.

3415 Kashiwa Street • Torrance, California 90505 • (213) 539-9140

Increase your computer's capacity for stability.

That Annoying Twitch

Marshall E. Smith
801 West Long Lake Road,
Apt. F-1
Bloomfield Hills, MI 48013

"Can you remedy the twitch in my video display?" asked my grandson. He had obtained an estimate for a professional repair job which included replacement of any doubtful ICs and a check of all solder connections. The size of the estimate meant a big hole in his cash reserves which he had planned to use for the purchase of peripherals. He remembered that I had done some do-it-yourself TV repairs and wondered if I could help him. With scant knowledge of computers, I reluctantly consented to tackle the problem.

Following the instructions in the *TRS-80 Microcomputer Technical Reference Handbook* we disassembled the case, trying not to strain the flexible connections and taking pains not to bend the logic board. We positioned the logic board with the devices up and the keyboard in an operable location, then plugged in the power supplies to the logic board and to the video display.

We turned on the power and checked the 12 volt, 5 volt and −5 volt supplies and found them satisfactory. I had learned that the computer had worked well except for the annoying horizontal twitch which appeared periodically at the left side of the screen when an image was being displayed. The problem seemed to be independent of duration of operation or temperature and would come and go without apparent cause. I connected my 3-inch oscilloscope and went to work.

With a display on the video, I connected the scope ground to the logic board ground, then did some careful probing among the IC pins, working back from the output jack. This was done carefully because I did not want to short-circuit or ground any live pins. All probing was done on the side of the logic board on which the devices are mounted.

With no twitch on the video I found a nice, steady pattern on pin 3 of Z49. Waiting for the inevitable twitch, which soon occurred, I probed the same pin and found the scope image was vibrating. I then probed other pins in the area and found to my pleasant surprise that the twitching stopped when the probe was on pin 2 or 3 of Z50. It was evident that there was a slight tendency towards instability in that circuit and that a small amount of capacitance to ground might cure it.

Sure enough, a 27pF ceramic type capacitor from pin 3 of Z50 to pin 15 (ground) of Z49 solved the problem (see Fig. 1). Care was exercised not to overheat the pins of the ICs during soldering. After several operational tests indicated success, we reassembled the computer. It has been working faithfully for several weeks now.■

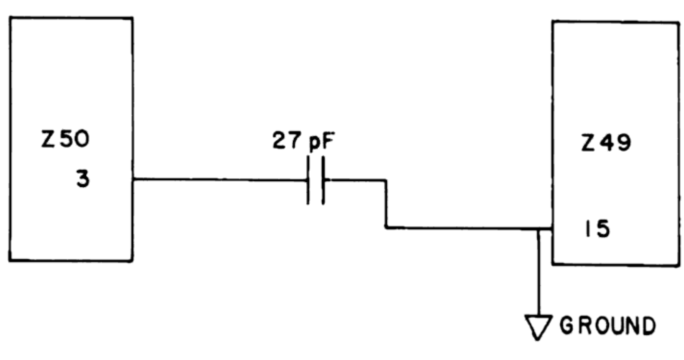


Fig. 1

COMPETENCY EXAM PREPARATION SERIES

This comprehensive set of programs consists of simulated exam modules, a thorough diagnostic package, and a complete set of instructional programs. It is designed to teach concepts and operations, provide drill and practice and assess achievement levels through pre and post testing. The Competency Exam Preparation Series provides a structured, sequential, curriculum encompassing mathematical, reading and writing instruction.

The C.E.P.S. program is designed for individual student use or use in a classroom setting. Programs provide optional printer capability, worksheet generation and performance monitoring. C.E.P.S. are available in two software formats.

National Proficiency Series	**$1,299.00**
N.Y.S. Regents Competency Test, Preparation Series	**$1,299.00**

If desired separate Mathematics and Verbal packages are available for $799.00 ea. A Spanish language version of the Mathematics Instruction Package is available at no extra charge.

COLLEGE BOARD PREPARATION SERIES 81/82

for TRS-80 NORTHSTAR™ PET, APPLE OSI

Each program confronts the user with a virtually limitless series of questions and answers. Each is based on past exams and presents material of the same level of difficulty and in the same form used in the S.A.T. Scoring is provided in accordance with the formula used by College Boards.

S.A.T., P.S.A.T., N.M.S.Q.T., set includes 25 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, and Mathematics. Price $149.95

EDUCATOR EDITION - includes all of the above programs plus detailed solutions and explanations. Price $229.95

Independent Tests of S.A.T. series performance show a mean total increase of 70 points in students' scores.

Update Pack to 81/82 specs. Available to previous owners. Price $69.95

ODYSSEY IN TIME

This spectacular adventure game adds a new dimension of excitement and complexity to **Time Traveler.** Players must now compete with the powerful and treacherous adversary in their exacting quest for victory.

To succeed they must vanquish this adversary in combat that rages across 24 time periods.

Odyssey In Time includes all the challenges of Time Traveler plus 10 additional eras, including those of Alexander the Great, Emperor Asoka of India, Attila the Hun, Genghis Khan. Each game is unique, and may be **interupted and saved** for later play.

available for APPLE & TR-80 PET, 32K - $39.95

ISAAC NEWTON

Perhaps the most fascinating and valuable educational game ever devised — ISAAC NEWTON challenges the players to assemble evidence and discern the underlying "Laws of Nature" that have produced this evidence. ISAAC NEWTON is an inductive game that allows players to intervene actively by proposing experiments to determine if new data conform to the "Laws of Nature" in question. Players may set the level of difficulty from simple to fiendishly complex.

In a classroom setting the instructor may elect to choose "Laws of Nature" in accordance with the complete instruction manual provided.

For insight into some of the basic principles underlying **ISAAC NEWTON** see GODEL, ESCHER, BACH by Douglas R. Hofstadler, Chapter XIX and Martin Gardner's MATHEMATICAL GAMES column in **Scientific American,** October, 1977 and June, 1959. $24.95.

TIME TRAVELER

Confronts players with complex decision situations and the demand for real time action. Using the **Time Machine**, players must face a challenging series of environments that include; The Athens of Pericles, Imperial Rome, Nebuchadnezzar's Babylon, Ikhnaton's Egypt, Jerusalem at the time of the crucifixion, The Crusades, Machiavelli's Italy, The French Revolution, The American Revolution, and The English Civil War. Deal with Hitler's Third Reich, Vikings, etc. At the start of each game players may choose a level of difficulty . . . the more difficult, the greater the time pressure. To succeed you must build alliances and struggle with the ruling powers. Each game is unique.

$24.95

Send $2.00 for complete Catalogue.

$5.00 Discount Coupon included in Catalogue.

PROGRAMS AVAILABLE FOR
TRS-80, APPLE II & PET
(unless otherwise indicated)

□ disk or □ cassette (please specify)

Krell Software Corp.

✓375

Send check or money order to
21 Milbrook Drive, Stony Brook, NY 11790
(516) 751-5139

All programs require 16K/TRS-80 programs require LEVEL II BASIC/APPLE programs require APPLESOFT BASIC

NY State Residents Add Sales Tax

An initial program loader for TRSDOS.

Through-Boot

Alan D. Smith
P.O. Box 119
Mortdale, N.S.W. 2223
Australia

One shortcoming of the TRS-80 disk system (TRSDOS) is the failure to provide an Initial Program Load (IPL) facility to take the user from power-up through to the desired Basic program without the need for numerous keyboard entries. The Auto Function with its one command limitation fails to solve the problem. Unless you happen to own, for example, Apparat's NEWDOS, you're stuck with a somewhat lengthy initialization sequence, particularly if loading I/O drivers. Out of such frustrations was born IPL/BAS.

IPL/BAS

IPL/BAS is a Basic program suitable for TRSDOS systems of 16K RAM and upwards. It allows the user to create a command file tailored to suit a particular IPL requirement. Each such file consists of a small machine language program plus the specified set of commands. For example, we might build a file called TAXIPL/CMD. Entering TAXIPL after power-up (or setting Auto TAXIPL) might: might:

- set Verify (ON);
- load an I/O driver PRINT/DVR;
- load and execute Basic;
- reply <ENTER> to How Many Files?;
- reply 49000 to Memory Size?
- and run program TAX/BAS. TAXPASS:1.

All this without any further entries from the keyboard!

Program Operation

To use IPL/BAS, key in the program and save to disk as IPL/BAS; be especially careful copying the data statements, lines 50 through 80. Whenever you want to set up an IPL command file, there are several steps to follow.

Execute Basic and reply 32511 to the Memory Size? query. (The command file is built in RAM starting at 32512 (7F00 hex), so that area needs to be protected. In fact, IPL/BAS will refuse to run if you haven't set the memory size correctly.)

Run IPL/BAS and when requested, key in the command set you require. Note that Enter should be typed as " + "; for example, to build the TAXIPL file outlined in the previous section, you would type:

```
VERIFY + LOAD PRINT/DVR + BASIC + +
49000 + RUN"TAX/BAS. TAXPASS:1" +
```

and then press Enter.

The program will return to TRSDOS, and you then write the command file to disk using the DOS Dump command. In the case in question we would enter:

```
DUMP TAXIPL/CMD (START = X'7F00',
END = X'7FBF',TRA = X'7F00')
```

The command file is now ready for use.

Program Listing 1.

```
7FØØ:  E5           PUSH  HL          Save HL
7FØ1:  2A 16 4Ø     LD    HL,(4Ø16)   Get original KB driver addr
7FØ4:  22 2B 7F     LD    (7F2B),HL   Save it
7FØ7:  21 11 7F     LD    HL,7F11     Get intercept address
7FØA:  22 16 4Ø     LD    (4Ø16),HL   Store in keyboard DCB
7FØD:  E1           POP   HL          Restore HL
7FØE:  C3 2D 4Ø     JP    4Ø2D        Return to TRSDOS
7F11:  DD E5        PUSH  IX          Save IX
7F13:  E5           PUSH  HL          Save HL
7F14:  2A 46 7F     LD    HL,(7F46)   Get delay count
7F17:  7C           LD    A,H
7F18:  B5           OR    L           Count zero?
7F19:  28 Ø4        JR    Z,7F1F      Branch if yes
7F1B:  2B           DEC   HL          Decrement delay count
7F1C:  AF           XOR   A           Return X'ØØ' to requestor
7F1D:  18 2Ø        JR    7F3F
7F1F:  DD 21 48 7F  LD    IX,7F48     Point IX to command set
7F23:  DD 7E ØØ     LD    A,(IX+ØØ)   Get next character
7F26:  CB 7F        BIT   7,A         Is it the last?
7F28:  28 ØA        JR    Z,7F34      Branch if no
7F2A:  21 ØØ ØØ     LD    HL,ØØØØ     Restore KB driver address
7F2D:  22 16 4Ø     LD    (4Ø16),HL
7F3Ø:  CB BF        RES   7,A         Reset hi-order bit of last
7F32:  18 ØE        JR    7F42
7F34:  21 25 7F     LD    HL,7F25     Update character pointer
7F37:  34           INC   (HL)
7F38:  FE ØD        CP    ØD          <ENTER>?
7F3A:  2Ø Ø6        JR    NZ,7F42     Branch if no
7F3C:  21 ØØ Ø5     LD    HL,Ø5ØØ     Reset delay count to maximum
7F3F:  22 46 7F     LD    (7F46),HL   Store updated count
7F42:  E1           POP   HL          Restore HL
7F43:  DD E1        POP   IX          Restore IX
7F45:  C9           RET               Return to requestor
7F46:  ØØ ØØ                          (Delay count)
7F48:                                 (Command set starts here)
```

Expensive Expansion

EXPENSIVE – The *LNW System Expansion II* comes with a full 32K of 200ns RAM, and built-in RS232c 20 MA current loop serial interface. That's for starters. Next, consider our heavy gauge steel case, power indicator lamp, gold-plated connectors, FR-4 glass epoxy circuit board with solder mask and silk screen legends. Then there is the parallel printer port, screen printer port, real time clock, and extra heavy duty onboard power supply with over current protection, over voltage protection and thermal shutdown. If that's not enough then there is the floppy disk controller, guaranteed operation at a 4MHz CPU speed and our 6 month warranty. Every one of these features is *STANDARD*. This is true system expansion. You get every 'expensive' feature without spending more.

CHEAP – Our price is $399.95. Any way you compare, features or price, LNW's *System Expansion II* is the clear winner. The LNW System has been field tested for over two years with thousands of users. It works with any DOS, is 100% TRS-80 Model I compatible and it works 'right out of the box'. If there is any doubt in your mind as to whether you should buy ours or the 'other guys', just ask an LNW owner!

WE ARE #1 – Number one in price, features, reliability, performance and delivery. LNW is committed to 'expensive' features and quality at reasonable prices. LNW is committed to support, thorough documentation, and reliability.

⟵53

This has made us the number one manufacturer of system expansion units and accessories for the Model I computer.

EXPANSION OPTION – 8-inch drive capability is as easy as plugging in the *LNDoubler 5/8* option*. Now you can have any combination of single- or double-density, single- or double-sided, 8"* and/or 5" disks on-line! 8-inch disk storage increased to 591,360 bytes – 77-track single-sided, double-density or 1,182,720 bytes – 77-track double-density, double-sided.

The *LNDoubler's* unique 5/8 switch allows you to boot from 5- or 8-inch system disks and it's accessible from outside the interface. The $219.95 *LNDoubler 5/8* comes with a double-density disk operating system (DOS+ 3.3.9), *complete* with BASIC and utility programs... ready to run your software.

Each of your present 40-track, single-sided 5-inch drives will store up to 184,320 bytes (formatted storage) – that's an 80% increase in storage capacity for only half the cost of just one disk drive. With three 8-inch double-density, double-sided drives your Model I will have 3.75 Megabytes of online storage – that's more storage than a Model II or Model III!

LNW Research Corp.

2620 WALNUT Tustin, CA. 92680
(714) 641-8850 (714) 544-5744

*8" drive operation requires special cable. 8" double-density requires 3.55MHz CPU speed up modification or LNW 80 4MHz computer

TRS-80 is a trademark of Tandy Corporation

```
10 CLS :CLEAR300 :DEFINT C,X,Y
20 QU$=CHR$(34) :MX$="MAXIMUM 120" :DE$="BASIC+++RUN"+QU$
30 IF PEEK(&H40B1) + PEEK(&H40B2)*256 < (&H7EFE) THEN 90 :REM CH
ECK MEMSIZE
40 PRINT@ 450, "** MEMORY SIZE NOT SET = 32511 OR LOWER  -  RUN
ABORTED **" :END
50 DATA 229,42,22,64,34,43,127,33,17,127,34,22,64,225,195,45,64,
221
60 DATA 229,229,42,70,127,124,181,40,4,43,175,24,32,221,33,72,12
7,221
70 DATA 126,0,203,127,40,10,33,0,0,34,22,64,203,191,24,14,33,37
80 DATA 127,52,254,13,32,6,33,0,5,34,70,127,225,221,225,201,0,0
90 FOR X=(&H7F00) TO (&H7F47) :READ Y :POKE X,Y :NEXTX :REM POKE
 MACH LANG PROG
100 Y=1 :GOSUB 380
110 PRINT "THIS PROGRAM CREATES AN AUTOMATIC DOS IPL ROUTINE , B
Y BUILDING"
120 PRINT "IT IN RAM BETWEEN 7F00 & 7FBF .......... THE DOS 'DUM
P' COMMAND"
130 PRINT "IS THEN USED TO OUTPUT IT TO DISK AS A COMMAND ( '/CM
D' ) FILE."
140 Y=13 :GOSUB 380
150 PRINT "USER MUST SUPPLY THE COMMAND SET TO BE USED AFTER BOO
TING DOS ;"
160 PRINT "ANY EMBEDDED <ENTER>'S SHOULD BE CODED AS '+'  -  FOR
 EXAMPLE ,"
170 PRINT "THE DEFAULT OPTION ( SUPPLIED IF YOU SIMPLY HIT <ENTE
R> ) IS :-"
180 PRINT TAB(22) DE$
190 Y=28 :GOSUB 380
200 PRINT "NOW TYPE COMMAND SET  ( "; MX$; " CHARACTERS - AS MAR
KED ) :-"
210 PRINT@ 768, CHR$(31);STRING$(120,".");"<--LIMIT"; :PRINT@ 76
8,;
220 LINEINPUT CS$ :IF CS$="" THEN CS$=DE$
230 CS=LEN(CS$) :IF CS<121 THEN CLS :PRINT@ 384,; :GOTO 290
240 FOR X=1 TO 8 :REM REMIND USER OF MAX 120
250     PRINT@ 664, STRING$(11," "); :GOSUB 280
260     PRINT@ 664, MX$; :GOSUB 280
270 NEXTX :GOTO 210
280 FOR Y=1 TO 150 :NEXTY :RETURN :REM DELAY LOOP
290 FOR X=1 TO CS
300     CH=ASC(MID$(CS$,X,1)) :IF CH=43 THEN CH=13 :REM "+"=<EN>
310     IF X=CS THEN CH=CH OR (&H80) :REM SET HI-ORD BIT ON LAST
320     POKE &H7F47+X,CH :REM POKE SUPPLIED COMMAND SET
330 NEXTX
340 PRINT "*** NOW TYPE THE FOLLOWING DOS COMMAND , EXACTLY AS S
HOWN :- ***";
350 PRINT "*** ( THE FILESPEC 'IPL/CMD' MAY BE VARIED AND/OR EXP
ANDED ) ***"
360 PRINT "DUMP IPL/CMD (START=X'7F00',END=X'7FBF',TRA=X'7F00')"
;
370 CMD"S" :REM RETURN TO TRSDOS
380 PRINT :FOR X=0 TO 127 :SET (X,Y) :NEXTX :RETURN
```

Program Listing 2

Points to Note

There are several points to remember about this program.

The maximum length of a command set is 120 characters—usually more than enough, but you can, if necessary, chain command files under DOS, i.e., have one invoke another.

You cannot use a command file to reply to Input, Line input, or INKEY$ statements contained within a Basic program.

When running IPL/BAS, simply pressing Enter without typing a command set provides a default of Basic+++run" (i.e. load and execute Basic, reply Enter to the How Many Files? and Memory Size? queries, and then type Run. This allows the user to manually key in the name of the program to be run).

For Assembler/Machine Language Buffs

Command files each contain a 72-byte program in Listing 1, which is POKEd into RAM between X'7F00' and X'7F47'. The specified command set is POKEd into the next 120 bytes (X'7F48' to X'7FBF'). The area above X'7FBF' is not used, because it is overwritten on entry to Basic on a 16K machine.

On initial entry, the keyboard driver address at X'4016'/X'4017' in the device control block is replaced by an intercept which effectively reroutes all subsequent keyboard requests to X'7F11'. The intercept routine at that address then passes the command set, one byte at a time, to the requestor (DOS or Basic). Before passing the last byte (identified by having bit seven set), the keyboard driver address is restored to its original value, thereby disabling the intercept and restoring the keyboard to normal operation. The delay loop prevents potential problems with functions that scan for Break during execution, as well as providing a little more time for you to observe what is happening! ■

A MAJOR BREAKTHRU FOR

TRS 80™*

Electronic Engineers
Electronic Hobbyists
Electronic Students
Ham Operators

A General DC-AC (steady state) Analysis of Any Circuit

Will Analyze and Compute:

- Node, Branch, Element, Voltages
- Node, Branch, Element, Currents
- Branch Power Dissipation
- Magnitude and phase values and complete frequency response with graphic display
- Modify any element in circuit for desired results

A complete operational manual supplied comparable to I.B.M.'s E.C.A.P.™ Program
*A.C. Analysis Program *$149.95*
*D.C. Analysis Program *$89.95*

To Order Write
G & L Software Enterp.
2304 N. 1st. Street
Upland, CA 91786

VISA   master charge

Do Not Send Cash in the Mail

TRS 80™ is a trademark of the Tandy Corp. E.C.A.P. is a trademark of International Business Machines, Inc.   ✓39

TRS-80 (TM)

MAKE A SOUND INVESTMENT IN YOUR MODEL I OR MODEL III LIBRARY

Why did Creative Computing call our MICROCOSM I "a worthwhile purchase"? What caused 80-US to say it "is exceptional"? Simple. BASICS AND BEYOND has produced consistently high-quality, creative, bug-free packages, each focusing on educational, practical and entertaining programs for your Model I or Model III TRS-80™ microcomputer. MICROCOSM I with 30 programs and MICROCOSM II with 20 programs are available for only $24.95 each.

With MICROCOSM III, BASICS AND BEYOND has broken the sound barrier. It includes "Seige", a medieval clash of arms, a serenade from "Computer Organ", "Pinball", which zings and bongs as it ricochets from flipper to bonuses, step by step learning with "Long Division", right wrong buzzers, and 16 more. MICROCOSM III is a 20 program package with sound for $24.95. This package available for Model I only.

BOX 10 • AMAWALK, N.Y. 10501   (914) 962-2355

BASICS & BEYOND, inc.   ✓49

No charge for postage or handling. N.Y. residents add 5% sales tax. TRS-80 is a trademark of Radio Shack, Division of Tandy Corp. Master Charge and Visa accepted.

WHATS THE BEST WORD???

How many times have you racked your brain for just the right word when writing an important letter, report, or article? How many times have your ideas been misunderstood because the words you used didn't express your thoughts clearly and accurately? How many times have your turned in a report or mailed an important letter and then hours later kicked yourself for not having thought of the best and most accurate words to use at the time you wrote it.

Now, by using the remarkable new **REFWARE THESAURUS** programs, your computer can speedily find those words that are on the tip of your tongue but that you can't quite remember at the moment. And it tells you how to spell them!

Just slip a **REFWARE THESAURUS** disk into your disk drive. Then type in your sentences or paragraphs. The computer will quickly offer a variety of alternatives, retyping your sentences or paragraphs with substitute possibilities chosen from its multi-thousand word vocabulary. It displays the revised sentences on your monitor or types them on your printer as you choose, so that you can mull them over and choose the one that most accurately expresses what you REALLY mean to say.

Having helped thousands of writers learn to express themselves with clarity as editor of such publications as the World Book Encyclopedia, the Encyclopedia Americana, and the Reader's Digest Almanac and Yearbook, David C. Whitney has drawn on decades of editorial experience to prepare the revolutionary **REFWARE THESAURUS** programs, bringing the speed and power of the computer to the aid of anyone who wishes to improve his writing or speaking.

In addition to the specific programs capable of substituting suggested alternate words for nouns and adjectives, **REFWARE THESAURUS Builder** enables engineers, physicians, lawyers, educators, business, physicists, chemists, and other professionals and specialists to develop their own individually tailored computer vocabularies of hard-to-remember technical words.

REFWARE THESAURUS Builder 1.0   Order No. 5001RT   **$149.95**

Series of eight utility programs enables the user to develop specialized computer thesaurus disk programs of hard-to-remember technical words and their alternates for personal use. Functions independently of Adjectives and Nouns programs.

REFWARE THESAURUS Nouns 1.0   Order No. 2001RT   **$39.95**

Draws on a vocabulary of 6,200 nouns to suggest alternates for the names of persons, places, things, and ideas that you use in your writing and speaking.

REFWARE THESAURUS Adjectives 1.0   Order No. 1001RT   **$39.95**

Utilizes 6,200 adjectives to assist you in choosing the most accurate modifiers to describe nouns and pronouns in your letters, reports, and speeches.

REFWARE THESAURUS User's Manual   Order No. 6001RT   **$9.95**

Complete description of use of each of the above programs. Included at no additional charge with each of program disks listed above. If ordered separately, price of documentation is refundable upon receipt of order and payment for program disk.

MINIMUM System Required TRS 80* Mod I or Mod III 48K with two disk drives.
*A Trademark of Tandy Corporation

REFWARE* Reference software division   David C. Whitney Associates, Inc.
P.O. Box 541, Chappaqua, N.Y. 10514   ✓374
*A trademark of David C. Whitney Associates, Inc.

Qty.	Order No.	Title	Unit Price	Total

Sales tax (N.Y. state residents only) ______
Postage and handling $ 2.00

Enclosed $ .... ( ) Check ( ) Money Order   TOTAL $
Bill ( ) Visa ( ) MasterCharge

Card # ....

Exp. Date .... Interbank # ....

Signature ....

Name .... Phone ( ) ....

Address ....

City .... State .... Zip ....
Mailed First Class, but allow 3 to 4 weeks for delivery, no C.O.D. orders.

✓See List of Advertisers on page 386

Will you have enough water when you need it most?

Firestream

George L. Gille
220 Clayton
Maryville, MO 64468

To operate effectively, fire hose must be supplied with the right amount of water under the correct pressure. The TRS-80 Level II Basic program described here was created to calculate the fire engine pumping pressure needed to effectively supply water to a wide variety of firestreams. (The firestream is the entire water supply system, including the water source, the fire engine pump, the fire hose and the nozzle.) I constructed the program as a training tool for volunteer firemen.

The Firestream

Firestreams are defined in terms of their volume of water flow in gallons per minute (GPM), and the operational pressure of the nozzle in pounds per square inch (PSI). Operational pressure is the velocity of the water flowing through the fire hose. The volume of water used is a function of the size of the fire hose and the pressure at which it is operating.

The type of fire hose nozzle used dictates the volume of water flow and the pressure which the firestream must supply. Pump operators must be able to adjust the outoing pressure of their pump to meet the requirements of the nozzles they are supplying with water. They are able to determine the correct pressure by knowing the characteristics of the firestream they are operating.

Nozzle Characteristics

Fog nozzles and solid stream nozzles are used in fire fighting. Fog nozzles produce fast moving jets of water that divide into drops with a definite cloud shape. They must be supplied with water pressure between 80 and 100 PSI. The volume of water put out by fog nozzles is determined by the design of the nozzle and the size of the fire hose to which it is attached.

The volume put out by solid stream nozzles may be calculated by the equation:

$$\text{Nozzle Output (GPM)} = 29.7 \cdot D^2 \cdot \sqrt{P}$$

D is the nozzle tip diameter in inches and P is the water pressure (PSI) at the nozzle. Solid stream nozzles usually operate at pressures between 40 and 60 PSI.

Pressure Losses

Pressure at the nozzle end of the firestream is always less than that measured at the pump, due primarily to these three factors:

- The difference in height between the pump and the nozzle;
- The water flow resistance produced by valves and stream splitters (wyes);
- The friction between the water flowing through the hose and the walls of the fire hose. The fire engine pump operator must be able to calculate these pressure losses to correctly set the output pressure of the pump.

Pressure loss due to a difference in height between the pump and the nozzle usually happens when a fire occurs in the upper floors of a building. The pressure loss is one PSI for each 2.3 feet the water is moved above the pump. When pumping the firestream downhill, there will be a one PSI increase in pressure for each 2.3 feet difference in height.

If the firestream is split into two or more smaller streams with the use of a wye connection a pressure loss of 5 PSI is assigned for each wye in the firestream.

To determine pressure losses due to friction the fire service uses 2½ inch diameter fire hose which is 100 feet long as the standard for calculation. To calculate the friction pressure losses in hoses with other diameters, determine the volume of water flow (Q) in hundreds of gallons per minute:

$$Q = \frac{\text{Nozzle Output in GPM}}{100}$$

The friction pressure loss per 100 feet of hose is then calculated by the equation:

$$FL = (2Q^2 + Q) \cdot HF$$

HF is the fire hose diameter adjustment factor and FL is the friction pressure loss in PSI. The HF would have a value of 1.0 when the friction loss for 2½ inch diameter fire hose is being calculated. The above equation is used when flow rates are greater than or equal to 100 GPM. If the flow rate in the firestream is less than 100 GPM, the equation for the calculation of the friction loss becomes:

$$FL = (2Q^2 + \tfrac{1}{2}Q) \cdot HF$$

This change in the equation is another adjustment for hose sizes less than 2½ inches in diameter.

Once the friction pressure loss for 100 feet of fire hose has been calculated, the total fric-

tion pressure loss may be determined by multiplying the calculated friction loss by the total length of fire hose in the firestream and dividing by 100. Note that the friction loss of each different diameter fire hose must be calculated separately.

The total pressure loss in the firestream is the sum of the pressure losses for friction, height differences, and wye connections. After making the above calculations, the pump operator sets the pump output pressure for a value equal to the nozzle operational pressure plus the total pressure loss in the firestream.

Sample Problem

A fire is located about 20 feet above the street on the third floor of a building. The fire engine is located at a fire hydrant 500 feet down the street. The firemen have used 600 feet of hose to reach the third floor. At the third floor the hose is split into two 150 foot long, 1½-inch diameter hoses, with a 1½ inch fog nozzle at the end of each of the hoses. The input into the computer program and the solution can be seen in Fig. 2.

This firestream would require a pumping pressure of 190 PSI to supply the two 1½ inch fog nozzles with a pressure of 100 PSI. You can see how this program simplifies the calculation of a proper pumping pressure in various situations. ■

Firestream Program Listing.

```
10 CLEAR 1000:CLS:PRINT STRING$(64,191)
20 DIM H$(7),HF(7),N$(5),NV(5,2),HT(3,2),CT(3,2)
30 PRINT TAB(21);"VOLUNTEER FIRE FIGHTER":PRINT:PRINT TAB(18);"*
* FIRE STREAM HYDRAULICS **"
40 FOR I=1 TO 7:READ H$(I),HF(I):NEXT I
50 FOR I=1 TO 5:READ N$(I),NV(I,1),NV(I,2):NEXT I
60 PRINT:PRINT STRING$(64,191):PRINT TAB(31);"BY":PRINT TAB(24);
"GEORGE L. GILLE":PRINT TAB(23);"VOLUNTEER FIREMAN":PRINT TAB(22
);"MARYVILLE, MISSOURI":PRINT:PRINT STRING$(64,191)
70 FOR I=1 TO 700:A$=INKEY$:NEXT I
80 CLS:PRINT"THE METHODS OF FIRE STREAM HYDRAULICS CALCULATIONS
AND DATA IN":PRINT"THIS PROGRAM WAS TAKEN FROM:":PRINT
90 PRINT"INTERNATIONAL FIRE SERVICE TRAINING ASSOCIATION":PRINT"
MANUAL NUMBER 201"
100 PRINT"FIRE SERVICE PRACTICES FOR VOLUNTEER FIRE DEPARTMENTS"

110 PRINT"FIFTH EDITION: 1971":PRINT:GOSUB 730
120 CLS:PRINT"PROGRAM DISCRIPTION:":PRINT"THIS PROGRAM CALCULATE
S THE ENGIN PRESSURE NECESSARY TO ALLOW"
130 PRINT"A FIRE STREAM TO OPERATE PROPERTLY. THE PROGRAM ASSUME
S THAT":PRINT"THE FIRE STREAM IS SET UP AS A REVERSE LAY, WITH T
HE ENGIN AT"
140 PRINT"THE ENGIN AT THE HYDRANT PUMPING WATER TO THE FIRE SCE
EN OR":PRINT"PUMPING FROM A RESERVE TANK.":PRINT
150 PRINT"YOU ENTER THE TYPE OF FIRE STREAM, STARTING AT THE PUM
PING":PRINT"ENGIN AND ENDING AT THE NOZZLE. THE FIRE STREAM MAY
BE SPLIT"
160 PRINT"WITH THE USE OF A WYE.":PRINT
170 PRINT"ALL WYED FIRE STREAMS ARE ASSUMED TO END IN THE SAME T
YPE OF ":PRINT"NOZZLE":GOSUB 730
180 CLS:PRINT"ENTER THE DIFFERENCE IN ELEVATION BETWEEN THE ENGI
N AND THE":PRINT"NOZZLE OR NOZZLES IN FEET."
190 PRINT:PRINT"NOTE: YOU CAN ASSUME EACH STORY OF A BUILDING IS
 10 FEET.":PRINT
200 INPUT"DIFFERENCE IN ELEVATION <FEET> =";EL:EL=EL/2.3
210 K=1:CLS:PRINT"NOW WE BEGIN TO ENTER THE NATURE OF THE FIRE S
TREAM.":PRINT
220 PRINT"NOTE: THE FIRE STREAM IS RESTRICTED TO TWO WYE CONECTI
ONS WITH":PRINT"A MAXIMUM OF THREE PROGRESSIVE HOSE SIZES."
230 PRINT:PRINT"LET'S BEGIN!!!":PRINT:GOSUB 730
240 CLS:PRINT"ENTER THE TYPE OF HOSE:":PRINT:PRINT"NUMBER TYPE O
F HOSE":PRINT STRING$(64,140)
250 IF K=1 THEN 280
260 FOR I=1 TO (K-1):PRINT (HT(I,2)*100);" FEET OF ";H$(HT(I,1))
;" HOSE ALREADY ENTERED":NEXT I
270 PRINT STRING$(63,140)
280 FOR I=1 TO 7:IF K=1 THEN 320
290 FOR J=1 TO 3
300 IF HT(J,1)=I THEN 330
310 NEXT J
320 PRINT"  ";I;" = ";H$(I);" HOSE"
330 NEXT I
340 PRINT STRING$(63,140):INPUT"ENTER THE NUMBER HOSE TYPE: ";HT
(K,1):PRINT:IF HT(K,1)>7 OR HT(K,1)<1 THEN PRINT"**** ERROR ****
":GOSUB 730:GOTO 240
350 IF K=1 THEN 390
360 FOR L=1 TO K-1:IF HT(L,1)>HT(K,1) THEN 380
370 PRINT"YOU ARE INCREASING HOSE SIZE!!!!!":PRINT:PRINT"PLEASE
ENTER AGAIN.":GOSUB 730:GOTO 240
380 NEXT L
390 PRINT"ENTER THE FEET OF ";H$(HT(K,1));" HOSE":PRINT"IN THE F
IRE STREAM.":PRINT
400 INPUT"FEET OF HOSE = ";HT(K,2):HT(K,2)=HT(K,2)/100
410 CLS:PRINT"ENTER TYPE OF CONNECTOR AT END OF THE ";H$(HT(K,1)
);" HOSE:":PRINT
420 PRINT"NUMBER TYPE OF HOSE CONNECTOR":PRINT STRING$(64,140)
430 FOR I=1 TO 5:PRINT"  ";I;" = ";N$(I):NEXT I
440 PRINT STRING$(64,140):INPUT"ENTER THE NUMBER OF THE CONNECTO
R TYPE ";CT(K,1):IF CT(K,1)>5 OR CT(K,1)<1 THEN PRINT"**** ERROR
```

Program continues

✓See List of Advertisers on page 386

Program continued

```
 ****":GOSUB 730:GOTO 410
450 IF K=3 AND CT(K,1)=5 PRINT"MUST END IN A NOZZLE":PRINT"PLEAS
E ENTER AGAIN":GOSUB 730:GOTO 410
460 IF K=3 THEN 480
470 IF CT(K,1)=5 THEN K=K+1:CLS:PRINT"ENTER NEXT HOSE":GOSUB 730
:GOTO 240
480 IF CT(K,1)<>4 THEN 550
490 CLS:PRINT"PLEASE ENTER THE REQUESTED CHARACTERISTICS OF THE
SOLID STREAM":PRINT"NOZZLE ON THE ";H$(HT(K,1));" HOSE.":PRINT
500 INPUT"ENTER THE OPERATION PRESSURE OF NOZZLE IN P.S.I.";P
510 IF P>90 OR P<50 THEN PRINT"*** ERROR *** PRESSURE OUT OF USU
AL OPERATION RANGE ***":PRINT"PLEASE RECONSIDER":GOSUB 730:GOTO
490
520 INPUT"ENTER TIP DIAMETER IN INCHES AS A DECIMAL";D
530 P1=SQR(P):GP=29.7*D*D*P1:PRINT"NOZZLE FLOW RATE = ";GP;" G.P
.M.":GOSUB 730
540 GOTO 560
550 P=NV(CT(K,1),1):GP=NV(CT(K,1),2)
560 WP=5*(K-1):Q=GP/100
570 G1=GP:GP=GP*((K-1)*2):IF GP<=0 THEN GP=G1
580 FF=1:IF GP<100 THEN FF=0.5
590 FOR I=1 TO K
600 CT(I,2)=HF(HT(I,1))*((2*Q*Q)+FF*Q)*HT(I,2)
610 TL=TL+CT(I,2)
620 NEXT I
630 EP=P+TL+WP+EL
640 CLS:PRINT"FOR OPERATION OF A ";N$(CT(K,1));" NOZZLE ":PRINT"
AT ";P;" P.S.I.":PRINT"NOZZLE FLOW = ";G1;" GPM":PRINT"FIRE STRE
AM FLOW = ";GP;" GPM"
650 PRINT"ENGIN PRESSURE REQUIRED = ";EP;" P.S.I.":PRINT:PRINT"F
RICTION LOSSES:":PRINT STRING$(63,140)
660 PRINT"  ELEVATION LOSS = ";EL;" P.S.I."
670 PRINT"  WYE LOSS = ";WP;" P.S.I."
680 FOR I=1 TO K
690 PRINT"  ";(HT(I,2)*100);" FT. OF ";H$(HT(I,1));" = ";CT(I,2)
;" P.S.I."
700 NEXT I
710 GOSUB 730
720 GOTO 830
730 INPUT"<PRESS ENTER>";Q$:RETURN
740 DATA "1 INCH RUBBER",91,"1-1/2 INCH RUBBER LINED",13
750 DATA "2-1/2 INCH RUBBER LINED",1,"3 INCH RUBBER LINED",0.385

760 DATA "4 INCH RUBBER LINED",0.09,"4-1/2 INCH RUBBER LINED",0.
051
770 DATA "5 INCH RUBBER LINED",0.031
780 DATA "1 IN. HOSE FOG NOZZLE",50,30
790 DATA "1-1/2 IN. HOSE FOG NOZZLE",100,100
800 DATA "2-1/2 IN. HOSE FOG NOZZLE",100,250
810 DATA "SOLID STREAM NOZZLE",0,0
820 DATA "WYE",5,0
830 CLS:PRINT"MEMORY = ";MEM:END
```

✓See List of Advertisers on page 386

With sound yet!

Hang Person

Photo 1.

Nathan Hilton
669 West Kitchen
Port Neches, TX 77651

Tired of squeaky chalk boards and dusty fingers when playing hangman? Replace those noisy chalk boards with a CRT and far-out computer sounds. Now you and your computer can enjoy each other's company playing Hang-80 on the privacy of your own cathode ray tube.

Hang-80 gets its sound from a utility called Super Sound by David G. Morr (May 1980, *80 Microcomputing*). This utility is easy to use and, with a little imagination, makes great sounds.

Sound

You need a small audio amplifier to hear the sound which comes out of the cassette output plug of your TRS-80 microcomputer.

Before you do any programming, answer your computer's memory size question with 32737 (16K machines). This allows Super Sound to be loaded into the top of RAM undisturbed.

Super Sound is found in lines 60–80. Line 60 sets the USR address. Line 70 reads and POKEs the data in line 80 into the top of RAM. Line 80 is the data for the Super Sound utility. Every time you see a USR( ), a sound will be produced.

The Program

With the exception of a gap here and there, you can Auto all the way through the program, making program typing a little easier. Furthermore, after you type Run and hit Enter, you can forget the Enter key because you won't need it during the game.

Here is how Hang-80 works: In line 100, variables A through Z are defined as integers. Lines 110 and 120 create the opening block letters that are assigned to A$. Lines 130–200 create letters and sounds.

Lines 3110–3140 are data lines containing the instructions. The loop in line 240 reads the data so lines 260 and 270 can conveniently read and print the instructions with flying sounds.

From 610 we GOSUB 2000. This is where the computer picks the word for you to guess at. Then we return and GOSUB 5000.

Program Listing

```
10 ' HANG-80
20 ' BY: NATHAN HILTON
30 ' USES `SUPER SOUND' BY DAVID MORR
40 ' SET MEM SIZE TO 32737 -16K RAM
50 ' SUPER SOUND IS LINES 60-80, THEN "HANG-80"
60 CLS:AD=32738:HI=INT(AD/256):POKE16527,HI:POKE16526,AD-HI*256
70 FORI=AD TO AD+28: READ DT: POKE I,DT: NEXT
80 DATA 205,127,10,62,1,14,0,237,91,61,64,69,47,230,3,179,211,25
5,13,40,4,16,246,24,242,37,32,241,201
90 CLEAR300
100 DEFINTA-Z:DEFSTRZ:DIMSK(15):Q=256:RANDOM
110 Q$=CHR$(191):S$=" ":A$=Q$+STRING$(2,176)+Q$+S$+Q$+STRING$(2,
179)+Q$+S$+Q$+STRING$(2,131)+Q$+S$+Q$+CHR$(131)+STRING$(2,179)+S
$+STRING$(4,176)+S$+CHR$(186)+CHR$(179)+CHR$(187)+CHR$(144)+CHR$
(170)+STRING$(2,131)+CHR$(171)+STRING$(31,32)
120 A$=A$+Q$+"  "+Q$+S$+Q$+"  "+Q$+S$+Q$+"  "+Q$+S$+Q$+STRING$(2
,176)+Q$+STRING$(6,32)+CHR$(181)+CHR$(176)+CHR$(184)+CHR$(149)+C
HR$(170)+STRING$(2,176)+Q$
130 B$="Welcome to :"
140 PRINT@460,"";:FORI=1TO12:L=L+5:PRINTMID$(B$,I,1);:U=USR(Q+L)
:FORJ=1TO20:NEXTJ,I
150 FORI=1TO2:U=USR(Q*70+RND(30)+10):PRINT@460,STRING$(12,32):FO
RJ=1TO200STEP5:T=USR((U/70)+J):NEXTJ:PRINT@460,B$:NEXTI
```

Program continues

"You can count the words you enter yourself or let the computer count them for you."

Lines 5000–5070 simply set up the playing screen.

Lines 630 and 640 put the spaces on the screen. Lines 660 through 780 process the entries and put the letters in the mystery word.

The first 29 data elements are the machine code numbers for Super Sound. The next 514 data elements are the words the computer picks from for the player to guess. Finally, there are the 14 lines of instruction. There is a total of 557 elements of data.

There is an alternative to typing 557 chunks of data. Leave the first 29 data elements alone. However, you can do what you want with the next 514.

You can use a different number of words as long as you make some number changes in lines 240 and 2050.

The number 543 in line 240 must be changed to account for the number of words you enter into the data lines. Simply add your number of words to 29 and enter that number in place of 543.

In line 2050 change the RND (514) to read RND (your number of words) after the change.

Counting

You can count the words you enter yourself or let the computer count them for you. I found it easier to let the computer do the counting.

After entering the program, execute the following line: RESTORE:FOR A = 1 TO 29:READ D$: NEXT A (enter). This gets you past the Super Sound so the next Read statement will read the first vocabulary word. Now execute the following line and remember to use Shift @ to freeze the display when necessary: FOR A = 1 TO 1000:READ D$:PRINT A;D$: NEXT A.

Whenever you see "Welcome to Hang-80" pop up, hit the Break key to get out of the loop. Now look at the last word printed; it has a number to its left. That number is the number of words you now have in the computer's vocabulary.

How to Play

When Hang-80 is run, you must answer a few questions the computer will ask you. The playing screen will appear and you will have all the information you need to play the game. Printed at 797 you will see a row of spaces with a flashing arrow to the left of them. The spaces are actually a row of cursors. The number of the CHR$(95)s in the row is the number of letters in the word you are to guess at. All you do is hit a letter and the computer will put it in the row of spaces if the letter you hit is contained in the word. Otherwise it is counted as a wrong guess. You lose when you don't have any wrong guesses left.■

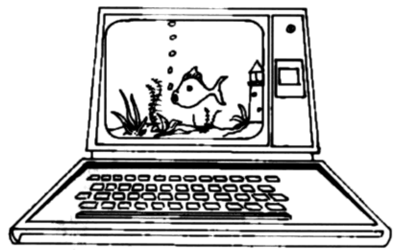
Program continued

```
160 FORI=1TO70STEP5:U=USR(Q*I+RND(30)+5):FORK=1TO50/I:NEXTK,I
170 PRINT@394,A$
180 FORI=255TO0STEP-5:F=F+5:U=USR(Q+I):U=USR(Q+F):NEXTI
190 F=0:FORL=1TO3:FORI=100TO1STEP-2:F=F+L:U=USR(Q+I):U=USR(Q+F):
NEXTI:F=0:NEXTL
200 FORG=25TO100STEP25:FORI=1TO30:U=USR(Q+G):NEXTI,G
210 RESTORE:PRINT@768,"Do you want instructions?"
220 W$=INKEY$:U=USR(Q+RND(5)+30):IFW$<>"Y"ANDW$<>"N"ANDW$<>"y"AN
DW$<>"n"THEN220ELSEIFW$="Y"ORW$="y"THEN230ELSE400
230 CLS:U=USR(Q*100+255)
240 RESTORE:FORA=1TO543:READD$:NEXT
250 FORA=20 TO 33
260 READD$:FORB=1TOLEN(D$):PRINTMID$(D$,B,1);:U=USR(Q+35+RND(3))
:FORG=1TO3:NEXTG,B:U=USR(Q+5)
270 PRINT:NEXTA
280 PRINT@960,"HIT <9> TO CONTINUE";
290 PRINT@965," ";:FORA=1TO5:Z=INKEY$:U=USR(Q+30):IFZ<>"9"THEN N
EXT ELSE 340
300 PRINT@965,"9";:FORA=1TO10:IFZ<>"9"THEN NEXT ELSE340
310 V=V+1:IF V>1000THENV=0:GOTO310: ELSE IF INT(V/4)=V/4 THEN 32
0ELSE 290
320 IF PEEK(16320)<>32 THENPRINT@960,"   ";:GOTO290:ELSE PRINT@9
60,"HIT";:GOTO290
340 CLS
400 K=640:PRINT:PRINT:PRINT:PRINT"Well you pick a  (1)- easy (re
litive)":PRINTTAB(17)"(2)- a little more difficult";CHR$(31)
410 FORI=1TO2:E$=INKEY$:IFE$<>"1"ANDE$<>"2"THENNEXTELSEIFE$="2"T
HENDF=0:GOTO500:ELSEDF=3:GOTO500
420 U=USR(Q+40):PRINT@K,CHR$(191);:K=K+1:IFK=1022CLS:PRINTCHR$(2
3)"Hurry up!":GOTO400ELSE410
500 IFDF=3 THEN G$="EASY" ELSE G$="NOT SO EASY"
610 CLS:GOSUB2000
620 CLS:GOSUB5000:G$=""
630 PRINT@797,"";:LG=LEN(W$):FORA=1TOLG:U=USR(Q+A*2)
640 PRINTCHR$(95);:NEXT
650 N=RND(3)+2+INT(LEN(W$)/2)+DF:M=0
660 IF LG=0 THEN7000:FORA=1 TO RND(10)+20:Z=INKEY$:IFZ=""THEN NE
XTELSE700
670 PRINT@785," ";:U=USR(Q+RND(5)+30)
680 FORA=1TO30:Z=INKEY$:IFZ=""THENNEXTELSE700
690 PRINT@785,CHR$(94);:U=USR(Q+RND(5)+30):GOTO660
700 IF (Z<"a" OR Z>"z") AND Z<>"9" THEN FOR A=100 TO 1 STEP-10:U
=USR(Q+A):NEXT:GOTO660:ELSEIFZ="9"THENCLS:U=USR(Q*75+44):CLS:RES
TORE:GOTO210
730 FOR A=1 TO LEN(G$) STEP 2: IF MID$(G$,A,1)<>Z THEN NEXT:N=N-
1: GOTO740:ELSE PRINT@528,STRING$(27,32);:U=USR(Q*127+255):PRINT
@528,CHR$(92);" LETTERS ALREADY GUESSED ";CHR$(92);:GOTO660
740 M=M+1
750 GOSUB1000:FOR A=1 TO LEN(W$):PRINT@860+A,Z;:FORK=1TO3:NEXTK:
PRINT@860+A," ";:IF NOT MID$(W$,A,1)=Z THEN NEXTA ELSE FORBB=1TO
5:PRINT@796+A," ";:PRINT@796+A,Z;:FORG=1TO5:U=USR(Q+ASC(Z)):NEXT
G:PRINT@796+A," ";:NEXTBB:PRINT@796+A,Z;:LG=LG-1:N=N+1:NEXTA
760 G$=G$+Z+" ":PRINT@428,N:IF N=0THENGOSUB4000: FORA=1TO3:PRINT
@385,STRING$(23,32);:U=USR(Q*100+50):PRINT@385,"# OF WRONG GUESS
ES LEFT";:U=USR(Q*100+50):NEXTA:FORY=47TO0STEP-2:FORX=0TO127STEP
9:SET(X,Y):NEXTX:U=USR(Q+Y):NEXTY:GOTO9000
770 PRINT@576,G$;
780 PRINT@300,M;:GOTO660
1000 FOR A=16157TO16157+LEN(W$):IF PEEK(A)=ASC(Z)THENGOTO660ELSE
 NEXT :RETURN
2000 CLS:FORA=1TO60STEP10:FORL=20TO1STEP-5:FORR=1TO8STEP2:U=USR(
Q+R+L+A):NEXTR,L,A:PRINT"Would you like the words":PRINTTAB(5)"(
1)- five letters and under":PRINTTAB(5)"(2)- six letters and ove
r":PRINTTAB(5)"(3)- both"
2010 U=USR(Q*10+100):FORA=1TO40:Z=INKEY$:IF Z<>"1"ANDZ<>"2"AND Z
<>"3" THENNEXTELSE2030
2020 U=USR(Q+50): FORA=1TO30:Z=INKEY$:IF Z<>"1"ANDZ<>"2"AND Z<>"
3" THENNEXT:GOTO2010:ELSE2030
2030 IFZ="1"THEN U=USR(Q*100+44):L=5 ELSE IFZ="2"THEN U=USR(Q*10
0+44):FORA=1TO20:NEXT:U=USR(Q*100+44):L=6 ELSE L=0:FORA=1TO3:U=U
SR(Q*100+44):FORB=1TO20:NEXTB,A
2040 CLS:RESTORE:FORA=1TO29:READD$:NEXT
2050 W=RND(514):PRINT@455,"ONE MOMENT PLEASE"
2060 FORA=1TOW:READ W$:U=USR(Q+A):NEXT
```

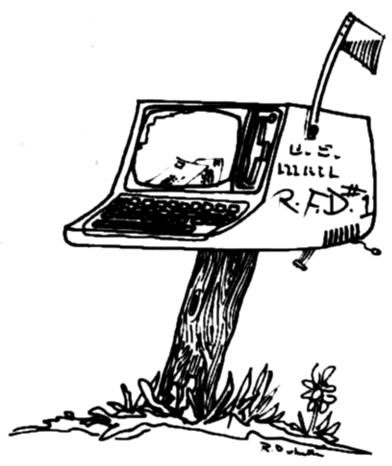
Program continues

101 USES FOR A DEAD MICROCOMPUTER

no.34



no.80



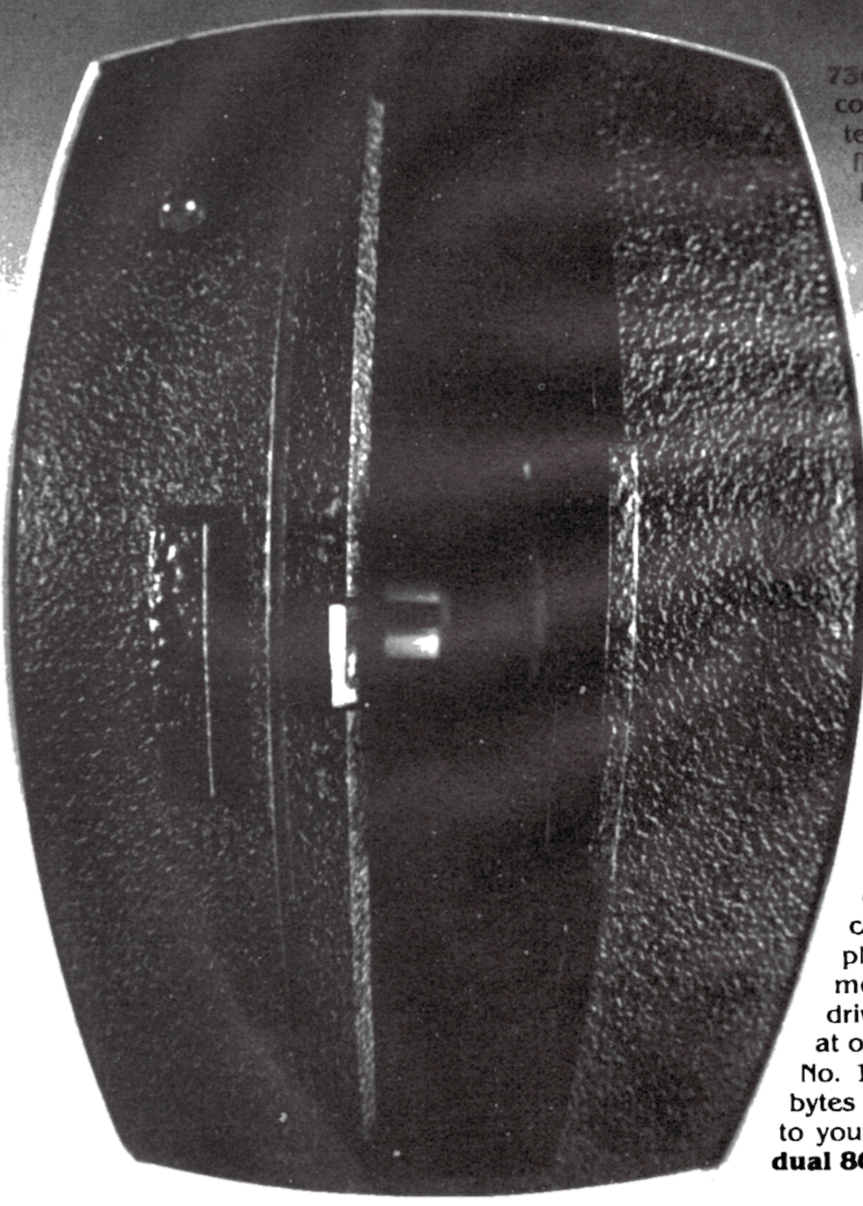
no.96

Program continued

```
2070 IF L=5 THEN IF NOT LEN(W$)<=5 THEN FORA=10TO1 STEP-1:U=USR(
Q+A):NEXT:GOTO2040
2080 IF L=6 THEN IF NOT LEN(W$)>=6 THEN FORA=10TO1 STEP-1:U=USR(
Q+A):NEXT:GOTO2040
2090 RETURN
2900 DATA actinium,aluminum,americium,antimony,argon,arsenic,ast
atine,barium,berkelium,beryllium,bismuth,boron,bromine,cadmium,c
alcium,californium,carbon,cerium,cesium,chlorine,shromium,cobalt
,copper,curium,dysprosium,einsteinium,erbium,europium
2910 DATA fermium,fluorine,francium,gadolinium,gallium,germanium
gold,hafnium,helium,holmium,hydrogen,indium,iodine,iridium,iron,
krypton,lanthanum,lawrencium,lead,lithium,lutetium,magnesium,man
ganese,mendelveium,mercury,molybdenum,neodymium,neon
2920 DATA neptunium,nickle,niobium,nitrogen,nobelium,osmium,oxyg
en,palladium,phosphorus,platinum,plutonium,polonium,potassium,pr
aseodymium,promethium,protactinium,radium,radon,rhenium,rhodium,
rubibium,ruthenium,samarium,scandium,selenium,silicon
2930 DATA silver,sodium,strontium,sulfur,tantalum,technetium,tel
lurium,terbium,thallium,thorium,thulium,tin,titanium,tungsten,ur
anium,vanadium,xenon,ytturbium,yttrium,zinc,zirconium
2940 DATA inconspicuous,inconstant,individually,indifference,inf
orm,inject,insufficient,insulate,invasion,jam,jail,jeep,jelly,ju
nk,knock,knot,knapsack,knowledge,know,known,koala
3000 DATA amazment,amandment,amoeba,amplify,amulet,amuse,adjust,
action,address,acknowledgement,accident,ace,apologize,bronco,bro
nze,bristle,brook,brood,bump,bee,bunt,cactus,caboose,caddie,cand
y,cap,capacity,cape,center,cellophane,celebration
3010 DATA centenial,clank,civilization,circus,clamber,clammy,cla
m,circulatory,citizenship,city,claim,clot,cloth,close,clock,clin
k,click,climb,clever,clothes,compound,complete,complaint,compile
,compassionate,compete,comprehension,cat,condor,coon
3020 DATA convent,coo,cook,cooler,cool,cookie,convert,crunch,cru
sh,crust,cuckoo,cue,cultivate,daunt,dead,deadly,dear,death,deal,
dean,dealt,delve,demand,delete,dice,different,difficult,dig,digg
er,discord,discolor,dirty,down,draft,drain,drag,eel
3030 DATA electric,electromagnet,electricity,ell,elk,elves,elf,e
lement,entry,enter,equal,exalt,extend,explode,extra,express,fan,
farewell,farm,fat,fate,fasten,father,fashion,fit,fist,flake,flag
,fission,fork,fussion,geophysical,genie,germ,get,good
3040 DATA habit,hailstone,hairpin,hall,halfway,hamster,hammock,h
ammer,handbag,handful,handle,happiness,hardware,harmony,harvest,
haunted,helicopter,hold,hypocritical,identification,idle,igloo,i
gnite,ill,imp,impact,imperial,impossibility,inch,indeed,indent

3050 DATA lump,lug,low,lurk,lull,luck,match,mop,material,melt,me
sh,minnow,mirror,mint,mystery,nag,new,news,never,nickel,open,ope
rate,ounce,other,our,oven,over,page,pale,palm,patch,patrol,peril
,perfect,perk,perplex,pig,picture,piano,pierce,pigoen,piece,pie
3060 DATA pliers,plow,plum,plug,plod,plot,ply,post,potent,potato
,probe,pro,proceed,professor,product,public,puff,pulp,puls,pulle
y,quick,quilt,quite,quicken,quiz,rabbit,race,ray,reach,realm,rea
l,read,refract,regular,request,run,rule,ruler,rug
3070 DATA rung,scan,scheme,scat,scare,scar,set,sea,setter,sew,sh
ock,shook,shoe,shot,short,shore,silk,signature,significance,simp
ly,simplify,skid,sketch,size,skate,sky,sole,soft,soda,softball,s
pook,splash,spoke,splatter,splinter,spontaneous,spoon
3080 DATA spoil,sport,stalk,stand,star,stamp,standard,stone,stor
m,stock,stocking,stockholder,stomach,stop,stork,stun,sub,subject
,subscribe,submit,submarine,sweat,sweep,sweet,swell,swim,swimmer
,swift,terror,terrific,terrible,than,thanks,timber
3090 DATA tomb,ton,told,tone,tooth,tonic,top,trailer,train,traff
ic,tramp,translate,translation,transformer,transistor,trip,trim,
trot,troop,triumphant,trouble,unbend,unbound,uncommon,uncomforta
ble,uncle,under,understand,uphold,upkeep,us
3100 DATA urban,verb,vein,watt,wave,way,wax,we,wear,weak,water,w
idth,wick,wig,will,wild,word,wonderful,wolf,wood,wooden,woodman,
woodcraft,wool,woolen,write,wrong,wry,xylophone,yam,yacht,yap,ya
rd,year,yarn,yellow,yong,zeal,zone,zero,zipper
3110 DATA "WELCOME TO HANG-80 ! !","  I will think of a word and
 you will guess it one"," letter at a time.","I will display som
e spaces that will represent the word you are to guess.","The nu
mber of spaces is the number of letters in the word."
3120 DATA "You simply hit a letter, and if it is in the word I",
" am thinking of, I will put it in the proper place"," in the ro
w of spaces."
3130 DATA "The spaces will have a flashing arrow pointing to the
m."," You will only have a certain number of wrong guesses to ma
ke,"," ELSE you loose! ! ! ! ! ! ! ! "," You win if you guess th
e word before all your"," wrong guesses are gone!"
3140 DATA"      - - - - - - - - - - GOOD LUCK - - - - - - - - - -"
4000 FORA=16157TO16157+LEN(W$):SK(A-16157)=PEEK(A):NEXT:RETURN
5000 PRINTTAB(8),A$;
5010 PRINT@155,G$;
```

Program continues

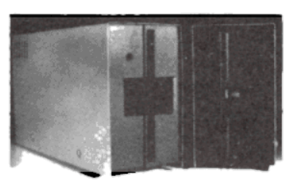
APPARAT OFFERS More bytes per buck!



730 K/bytes of storage. Apparat has combined its Newdos/80 operating system and a dual-sided 80 track mini-floppy drive to give you up to 733,440 bytes of storage in a single volume. Newdos/80 version 2.0 expands the capability of double density drives, so you'll have greater applications for your TRS-80® model I and III.

Drives plug directly into an expansion interface (requires installation of a double density controller) or the model III disk bus with our single volume cable so you can now have over 2 megabytes of storage on-line with standard mini-floppy diskettes. Each drive has up to 573 free grans, for a total of 1,719, on a maximum of three 80 track drives, which can be added to a TRS-80 model I. Model III's can have up to 4 dual 80's on-line (almost 3 megabytes).

These drives can "read" standard 35 or 40 track diskettes using Newdos/80 version 2.0 which will allow skipping every other track.



Drives come complete with case, power supply and documentation. The drives are priced at only **$515** (Cat. No. 1-705, specify I or III). Now, at 1424 bytes per buck, it just might be the answer to your storage problems. **Special — Two dual 80 track drives only $999.**

Apparat, Inc.

4401 So. Tamarac Parkway, Denver, CO 80237 (303) 741-1778

VISA

264

"ON GOING SUPPORT FOR MICROCOMPUTERS"

Program continued

```
5020 PRINT@257,"# OF GUESSES MADE",CHR$(94);
5030 PRINT@385,"# OF WRONG GUESSES LEFT",CHR$(94);
5040 PRINT@528,CHR$(92);" LETTERS ALREADY GUESSED ";CHR$(92);
5050 PRINT@768,"WORD TO GUESS --";CHR$(94);
5060 PRINT@960,"FOR NEW GAME HIT <9<";
5070 RETURN
7000 PRINT@797,STRING$(LEN(W$),32);:FORA=1TO20:U=USR(Q+RND(10)+2
5):NEXT
7010 PRINT@797,"";:FORA=1TOLEN(W$):PRINTMID$(W$,A,1);:FORL=1TO10
STEP5:FORT=20TO1STEP-5:U=USR(Q+T+L):NEXTT:U=USR(Q+L):NEXTL:U=USR
(Q+10+A):NEXTA
7020 IF V=1 THEN 7030 ELSE V=1:FORA=1TO2:PRINT@797,STRING$(LEN(W
$),32);:FORT=1TO20:U=USR(Q+T):NEXTT:PRINT@797,W$;:FORT=1TO20:S=U
SR(Q):NEXTT,A:GOTO7000
7030 CLS:Q$="You guessed the word, so................":FORA=1TOL
EN(Q$):PRINTMID$(Q$,A,1);:U=USR(Q+A+220):NEXT
7040 PRINT"I guess you win.":GOTO280
9000 CLS:FORA=1TO2:PRINT@390,A$;:FORT=4TO1STEP-1:FORG=1TO8STEP2:
FORY=20TO0STEP-10:U=USR(Q+Y+G+T):NEXTY:U=USR(Q+G+T+A):NEXTG:U=US
R(Q+T):NEXTT:PRINT@390,STRING$(LEN(A$),32);:FORS=1TO5:U=USR(Q+S+
A):NEXTS,A
9010 Q$="THE WORD WAS ------ "+W$:PRINT@450,"";
9020 FORA=1TOLEN(Q$):PRINTMID$(Q$,A,1);:U=USR(Q+20):FORS=1TO4:NE
XTS,A
9030 Q$="YOU LOSE AND - - - - - - - - - - - "+CHR$(162)+CHR$(183)+
" "+CHR$(181)+CHR$(180)+CHR$(149)+CHR$(149)+CHR$(151)+CHR$(171)
9040 FORA=128+LEN(Q$)TO128STEP-1:PRINT@A,MID$(Q$,A-127,1);:U=USR
(Q+A-50):NEXT
9050 PRINT@704,"";
9060 Q$="WHEN YOU MADE TOO MANY WRONG GUESSES,":FORA=1TOLEN(Q$):
PRINTMID$(Q$,A,1);:U=USR(Q+60):NEXT
9070 Q$="YOUR "+CHR$(34)+"PART WORD"+CHR$(34)+" LOOKED LIKE"+CHR
$(94)
9080 PRINT:FORA=1TOLEN(Q$):U=USR(Q+A):PRINTMID$(Q$,A,1);:NEXT
9090 FORA=16157TO16157+LEN(W$)
9100 POKEA,SK(A-16157):U=USR(Q+A):U=USR(Q+RND(20)):NEXT
9110 GOTO280
```

SAVE / on Software and Hardware for TRS-80®

SOFTWARE

NEWDOS/80 version 2.0(specify Model I or III) $149.00

Flextext I $29.95
Utilize the additional features of the 737/739 under Model I or III Scripsit®

Flextext II $29.95
Utilize the additional features of the NEC spinwriter under Model II Scripsit®

Flextext 80 (available soon)
Utilize the additional features of the MX-80 under Model I or III Scripsit®

Chextext $79.95
Spelling checker for Model I and III word processing

Microclinic $24.95
The ultimate in memory and disk diagnostics

Meal Master $24.95
Meal planning and grocery shopping aid

Sprint 80 $14.95
Creates your own custom serial printer driver

Professional Football Predictor $29.95
Let your TRS-80 pick the winners

Bionic Basic $74.95
Expand your Model I disk basic features

Interactive Business System $500.00
An inventory control system for manufacturing

Micro Accounting System $489.00
The finest general business package we've seen

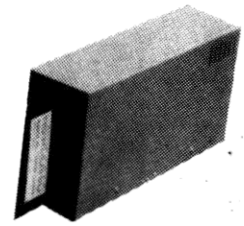
HELP/CMD $14.95
Adds "HELP" feature to NEWDOS/80 version 1.0

NEW LOWER PRICES ON TANDON DISK DRIVES

Complete with power supply, chassis & configured for TRS-80® Model I or III

- Single sided 40 track $295.00
- Dual 40 or Single 80 track $425.00
- Dual sided 80 track $515.00
- **Special** - 2 Dual sided 80's $999.00

Model I DOS and drive package add $54.00
Includes above drives and NEWDOS/80 version 1.0-the established DOS.



MODEL III SUPER SYSTEM

Approximately 2.1 megabates of disk storage using 2 dual headed 40 track drives internally and 2 dual headed 80 track drives externally.

$3299.00 **with NEWDOS/80 version 2.0**
$1895.00 **same as above with only 2 single sided 40 track drives(no DOS)**

NOTE: These Model III computers contain Apparat installed disk drives and memory. They are warranted by Apparat, Inc. for 90 days.
Attention Dealers: Call or write for information on our new dealer discount package.

MISCELLANEOUS

Apparat's Data Separator	$29.95
16K TRS 80 or APPLE memory kits	$19.95
9½" x 11" blank white,tractor feed paper,full box	$24.95
9½" x 11" blank white,tractor feed paper,half box	$14.95
14½" x 11" green bar,tractor feed paper,full box	$34.95
3½" x 15/16" 'tractor feed mailing labels	$19.95
Model II add on drives	(call for prices)
Model II business software	(call for prices)

SERVICE CENTER SPECIAL

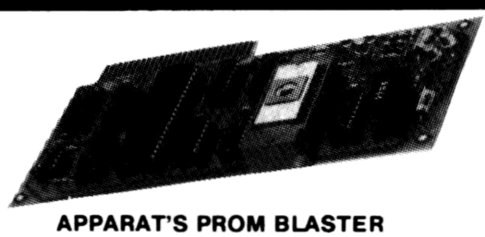
Disk drive tune-up $29.95
Complete-cleaning, alignment, and speed adjustment.
With any tune-up you may purchase your choice
1 box Verbatim diskettes $19.95
BASF head cleaner $9.95

PRINTERS

Call for our incredibly low prices on printers.
All printers come with free cable. We carry:
ANADEX CENTRONICS NEC
EPSON OKIDATA DAISY WHEEL

CABLES

Printer cables (specify printer and TRS-80 Model)	$29.95
Disk Drive Cables	
2 Drive	$29.95
3 Drive	$34.95
4 Drive	$39.95
Extender Cable	$12.95
Special Cables	(call for prices)



APPARAT'S PROM BLASTER

An eprom programmer for all 25 X X and 27 X X chips TRS-80® MOD I & III or APPLE $149.00
Bus Extender with power source, enclosure and 5 card slots $149.95
Mini version with 2 card slots and no enclosure $69.95

More Savings

MODEL I DOUBLE DENSITY PACKAGE

Converts a standard TRS-80 Model I to Double Density.
LN Doubler-NEWDOS/80 V.2.0 & box of 10 Verbatim Diskettes.
$330.00 value for only $269.00 • LN Doubler alone $149.00

DISKETTES

Double density-soft sectored-replacement guaranteed-Hard Hole protected. (5¼" only)

Verbatim Datalife 5¼" 40 track	$24.95
Apparat's No Name 5¼" 40 track	$21.95
Verbatim Datalife 8" Model II	$39.95

FLIP SORT SPECIAL

The ultimate file box filled with 50 diskettes 5¼" $119.95
Flip sort alone $29.95

VISA

Apparat, Inc. ✓264 "On-going Support for Microcomputers"

4401 S. Tamarac Pkwy. • Denver, CO 80237 • (303) 741-1778 • (800) 525-7674

Scripsit & TRS 80 are a registered trademark of Tandy Corporation
Freight F.O.B. Denver call for shipping charges. Foreign Orders shipped Air Freight

✓See List of Advertisers on page 386

Patches Plus for Level II Basic.

A Tale of Two Drivers

John T. Blair
122 Dumont Ave.
Norfolk, VA 23505

I have developed two printer driver patches, one for Level II Basic, operating under either TRSDOS or Basic, the other for NEWDOS's EDTASM. The patches have the following features: They output a line feed after a carriage return (to enable printers requiring this feature), provide for an automatic form feed after a given number of lines and enable the user to output a top-of-form from the keyboard. An additional feature, to be used with EDTASM, is a page header printed with the date and page number.

Hardware or software may be used to cause a line feed on a carriage return. Either solution will work equally well, but software is simpler and less expensive.

Program Listing 1 is a predriver (executed before ROM printer driver) for Level II Basic and TRSDOS or NEWDOS. This program is loaded into the upper end of memory and must be protected. To capture the printer driver, which is called from ROM by the various Level II and DOS print commands, change the address of the printer driver by modifying the printer Device Control Block (DCB). The DCB is located at 4025H–402CH, and the printer driver's Least Significant Byte (LSB) is at 4026H, the Most Significant Byte (MSB) is at 4027H.

By altering the printer driver address in the DCB, program control is diverted to the predriver, lines 27–61. The address in the DCB is changed by the initialization section of the program. (Note: Readers not using

Program Listing 1

```
                    00001 ;**********************************************************
                    00002 ;*                                                        *
                    00003 ;*                  P A T C H / T X T                     *
                    00004 ;*                                                        *
                    00005 ;*             PRINTER PATCH FOR <LF> AFTER <CR>          *
                    00006 ;*                        VERSION 1.0                     *
                    00007 ;* BY JOHN BLAIR WA4OHZ  & TOM THOMPSON                   *
                    00008 ;*                                                        *
                    00009 ;**********************************************************
                    00010
                    00011 ;                         INITIALIZATION
                    00012 ;       THIS SECTION LINKS THE ROM PRINTER DRIVER ROUTINE
                    00013 ; WITH THIS PROGRAM. ALSO THIS SECTION WILL GENERATE A
                    00014 ; PRITECTION OF THIS PROGRAM TO DOS AND BASIC. THEN
                    00015 ; RETURN TO DOS.
                    00016
FE00                00017        ORG     0FE00H          ;SOME DOS'S REQUIRE THE
                    00018                                ;TOP 128 BYTES OF MEMORY
                    00019
FE00                00020 INIT   EQU     $
FE00 210FFE         00021        LD      HL,START
FE03 22B240         00022        LD      (40B2H),HL      ;SET PROTECTED MEMORY
FE06 22D640         00023        LD      (40D6H),HL      ;       "
FE09 22BA0F         00024        LD      (4026),HL       ; LINK PATCH TO PRINT DCB
FE0C C32D40         00025        JP      402DH           ;RE-ENTER DOS
                    00026
                    00027 ;**********     PATCH     **********
                    00028
```

Program continues

disk should change line 22 to: LD(40B2)HL. This automatically sets protected memory for disk users. The address of the jump instruction in line 38 must be changed from 402DH to 1A19H, the warm start for Level II Basic.)

Pre-driver operation is quite simple: It checks the character to be printed. If it is not a carriage return, control is given back to Radio Shack's printer driver, located at 058DH. If the character is a carriage return, lines 42-45 output the line feed directly to the printer and increment the line counter. When the line counter indicates 54 lines have been printed, the pre-driver executes lines 53-58 to generate a form feed.

Willful Form Feeds

The last feature resets the line-per-page counter, and outputs a form feed at will. The shift @ sign, a printable character with no meaning as a Basic command, is used for this purpose. After a listing has finished, the last page is usually almost filled. Enter LPRINT shift @ to reset the counters and top-of-form and to get the page out of the printer. This keeps the printer's counters and the software counters in sync.

If your printer does not accept a form feed, 0CH, use the optional form feed routine shown in Program Listing 3. Those using Level II *must* protect memory on power up, then load the object program using the System call. After the program has loaded, enter a slash to execute the initialization that links the pre-driver to Basic.

Readers using disk should put this in the automatic boot file on disk. This will cause the line feed on the carriage return patch to be booted every time the DOS is booted. You may now use all DOS and Basic commands that send output to the printer.

Page Headers for EDTASM

Program Listing 2 is a modification of Apparat's NEWDOS Plus EDTASM. The program provides: a line feed on carriage return; a way of resetting the counts and giving a form feed; a

Program continued

```
                00029 ;         THIS SECTION IS A PRE-DRIVER THAT ALLOWS A
                00030 ; LINE-FEED UPON DETECTION OF A CARRIAGE RETURN. IT
                00031 ; ALSO WILL GENERATE A FORM-FEED AFTER THE DETECTION OF
                00032 ; THE 54TH CARRIAGE RETURN. A SPECIAL CHARACTER OF A
                00033 ; SHIFT `,  USING THE LPRINT "<SHIFT> @" COMMAND FROM
                00034 ; BASIC WILL RE-SET ALL COUNTERS AND GIVE A TOP-OF-FORM.
                00035
FE0F            00036 START EQU     $
FE0F 79         00037       LD      A,C             ;GET CHAR TO BE PRINTED
FE10 FE60       00038       CP      60H             ;IS IT A SHIFT ` ?
FE12 2822       00039       JR      Z,FRMFED        ;OUTPUT <FF>
FE14 FE0D       00040       CP      0DH             ;IS IT  <CR> ?
FE16 C28D05     00041       JP      NZ,058DH        ;NO  GOTO PRINTER DRVR
FE19 CDD105     00042 LF    CALL    05D1H           ;PRINTER STATUS CHECK
FE1C C219FE     00043       JP      NZ,LF
FE1F 3E0A       00044       LD      A,0AH           ;SET OUTPUT CHAR = <LF>
FE21 32E837     00045       LD      (37E8H),A       ;OUTPUT CHAR
FE24 DD7E04     00046       LD      A,(IX+04)       ;GET LINE COUNTER
FE27 FE35       00047       CP      53D             ;54 LINES YET ?
FE29 C245FE     00048       JP      NZ,EXIT         ; NO.  OUTPUT <CR>
FE2C CDD105     00049 CR    CALL    05D1H           ;GET PRINTER STATUS
FE2F 20FB       00050       JR      NZ,CR
FE31 3E0D       00051       LD      A,0DH           ;OUTPUT CHAR = <CR>
FE33 32E837     00052       LD      (37E8H),A       ;OUTPUT LAST <CR>
FE36 CDD105     00053 FRMFED CALL   05D1H           ;PRINTER STATUS ?
FE39 20FB       00054       JR      NZ,FRMFED
FE3B 3E0C       00055       LD      A,0CH           ;SET OUTPUT CHAR = <FF>
FE3D 32E837     00056       LD      (37E8H),A       ;OUTPUT CHAR
FE40 DD360400   00057       LD      (IX+04),00      ;RESET LINE COUNTER
FE44 C9         00058       RET
                00059
FE45 C38D05     00060 EXIT  JP      058DH           ;OUTPUT <CR>
FE00            00061       END     INIT
00000 TOTAL ERRORS
```

```
     00100 ;*****************************************************
                00110 ;*                                                        *
                00120 ;*              OPTIONAL  'FORM FEED' ROUTINE             *
                00130 ;*                                                        *
                00140 ;**********************************************************
                00150
                00160
                00170 ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                00180 ;$                                                        $
                00190 ;$      NOTES:                                            $
                00200 ;$      1)   THIS CAN REPLACE THE SECTION OF THE          $
                00210 ;$        FORM FEED ROUTINE THAT OUTPUTS THE 0CH.         $
                00220 ;$      2)   THE REFERENCE COUNTER (IX+03) IS 67          $
                00230 ;$       LINES / PAGE. THIS CAN BE CHANGED BY             $
                00240 ;$      POKING THE VALUE DESIRED, OR BY ADDING:           $
                00250 ;$      LD      HL,4028H                                  $
                00260 ;$      LD      (HL),XX                                   $
                00270 ;$      WHERE XX = THE # OF LINES/PAGE DESIRED            $
                00280 ;$      INTO THE INITIALIZATION ROUTINE                   $
                00290 ;$                                                        $
                00300 ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                00310
                00320
0000 DDE5       00330         PUSH    IX
0002 CD9F05     00340         CALL    059FH          ;FINE # LINES LEFT ON
                00350                                ;PAGE, AND OUTPUT AS
                00360                                ;<LF>'S.
0005 DD3405     00370         INC     (IX+05)        ;INC PAGE COUNTER
0008 DDE1       00380         POP     IX             ;INC PAGE COUNTER
                00390
0005            00400         END     ;  ***  DELETE THIS LINE  ***
00000 TOTAL ERRORS
```

Program Listing 2

```
00100 ;****************************************************
00110
00120 ;*          A S M P T C H / T X T                     *
00130
00140 ;*     PRINTER PATCH FOR <LF> AFTER <CR>               *
00150 ;*               AND  <FF>                             *
00160 ;*          FOR NEWDOS'S DISK BASED EDTASM             *
00170 ;*                                                     *
00180 ;*  BY  JOHN BLAIR  &  TOM THOMPSON                    *
00190 ;****************************************************
00200
00210 ;************  DEFINATION OF LABLES ******************
```

Program continues

method to title and number a page; and inserts the date in the header.

This program will be added to the end of EDTASM. All references to the beginning of the text buffer must be changed to the new beginning of the buffer, the function of lines 350–600.

Generating a line feed after a carriage return is easiest. Since NEWDOS makes use of a ROM call to access the printer driver, we need only change the address of the printer driver by modifying the printer Device Control Block (DCB). This is done in the initialization section. The printer driver's address is altered and program control is diverted to the pre-driver, lines 610–1840. This section of the program checks the character to be printed to determine if it is a carriage return. If not, control is given back to ROM's printer driver located at 058DH. If the character to be printed *is* a carriage return, the pre-driver outputs the character: lines 850–910 output the carriage return directly to the printer. Lines 930–1040 output the line feed, increment the line counter and check to see if 54 lines have been printed. If not, the character to be printed (carriage return 0DH) is restored, and the pre-driver returns control to EDTASM. When 54 lines have been printed, the pre-driver executes lines 1060–1150 to generate a form feed.

There are two methods of generating a form feed. The first has the printer accept the ASCII form feed command, a 0CH. The pre-driver outputs a 0CH directly to the printer, then resets the line counter, increments the page counter and calls the routine to print the page headers. The second method is to call the form feed routine from ROM, which outputs line feeds by replacing lines 1090–1150 with the alternate form feed routine (Program Listing 3).

Printing the page header information is more complex (lines 1380–1840). First, the header message is printed by the OUTSTR subroutine, then the page number is printed. The OUTSTR routine requires that the HL register contain the ad-

Program continued

```
                 00220
2B00             00230 OFFSET   EQU      2B00H
7480             00240 APPMSG   EQU      9F80H-OFFSET      ;START OF APPARAT SIGN ON
8000             00250 ASMST    EQU      5500H+OFFSET      ;START OF RELOC. ASM2
4029             00260 LINCNT   EQU      4029H             ;LINE/PAGE COUNTER
7481             00270 MSG1     EQU      APPMSG+1
402A             00280 PAGCNT   EQU      402AH             ;PAGE # COUNTER
058D             00290 PRINT    EQU      058DH             ;ROM PRINTER DRIVER
37E8             00300 PRTR     EQU      37E8H             ;PRINTER ADDRESS
05D1             00310 STATUS   EQU      05D1H             ;ROM PRINTER STATUS CK.
0033             00320 VIDEO    EQU      0033H             ;ROM VIDEO DISPLY
58DA             00330 WARMST   EQU      58DAH             ;REENTRY TO ASM2
                 00340
                 00350 ;** REPLACE START OF BUFFER IN EDTASM WITH NEW BEG **
                 00360
                 00370 ;        THIS SECTION REPLACES ALL EDTASM REFERENCES TO
                 00380 ; THE BEGINNING OF TEXT BUFFER WITH THE BEGINNING OF THE
                 00390 ; NEW BEGINNING OF TEXT BUFFER.
                 00400
8118             00410          ORG      8118H
                 00420
8118 21B876      00430          LD       HL,DONE-OFFSET
838A             00440          ORG      838AH
838A 21B876      00450          LD       HL,DONE-OFFSET
8410             00460          ORG      8410H
8410 11B876      00470          LD       DE,DONE-OFFSET
8707             00480          ORG      8707H
8707 11B876      00490          LD       DE,DONE-OFFSET
87DB             00500          ORG      87DBH
87DB 21B876      00510          LD       HL,DONE-OFFSET
8850             00520          ORG      8850H
8850 21B876      00530          LD       HL,DONE-OFFSET
8A39             00540          ORG      8A39H
8A39 21B876      00550          LD       HL,DONE-OFFSET
8A80             00560          ORG      8A80H
8A80 21B876      00570          LD       HL,DONE-OFFSET
8F27             00580          ORG      8F27H
8F27 21B876      00590          LD       HL,DONE-OFFSET
                 00600
                 00610 ;****************  PRINTER  PRE-DRIVER  *****************
                 00620
                 00630 ;        THE PRE-DRIVER IS CALLED INSTEAD OF THE ROM
                 00640 ; PRINTER DRIVER. IT CHECKS FOR A <CR>, AND THEN OUTPUTS
                 00650 ; A <LF>. IF CHARACTER TO BE PRINTED IS NOT A <CR> THEN
                 00660 ; THE ROM PRINTER DRIVER IS USED.
                 00670
A13A             00680          ORG      0A13AH            ;OVERLAY TO RELOC. ASM2
                 00690
                 00700 ;        THIS ADDITION TO EDTASM IS TAIL ENDED TO THE
                 00710 ; PROGRAM. HOWEVER, EDTASM HAS BEEN BLOCKED MOVED TO
                 00720 ; START AT 8000H WITH LMOFFSET. EXECUTION OF EDTASM
                 00730 ; WILL BLOCK MOVE IT DOWN TO IST NORMAT ORIGION OF 5500H.
                 00740
                 00750 ;>>>>>>>>>>     OUTPUT ALL CHAR  EXCEPT  <CR>  <<<<<<<<<<
                 00760
763A             00770 START    EQU      $-OFFSET
A13A 79          00780          LD       A,C               ;GET CHAR TO BE PRINTED
A13B D9          00790          EXX
A13C CD3300      00800          CALL     VIDEO
A13F D9          00810          EXX
A140 FE0D        00820          CP       0DH               ;IS IT A <CR>
A142 C28D05      00830          JP       NZ,PRINT          ;JP TO ROM PRINTER DRIVER
                 00840
                 00850 ;>>>>>>>>>>     OUTPUT  <CR>     <<<<<<<<<<
                 00860
7645             00870 OUTPUT   EQU      $-OFFSET
A145 CDD105      00880          CALL     STATUS            ;CHECK THE PRINTER STATUS
A148 C24576      00890          JP       NZ,OUTPUT         ;IS IT READY?
A14B 79          00900          LD       A,C               ;RESTORE <CR>
A14C 32E837      00910          LD       (PRTR),A          ;OUTPUT <0D> TO PRINTER
                 00920
                 00930 ;>>>>>>  OUTPUT  <LF>  AND  CHECK LINE COUNTER  <<<<<<
                 00940
764F             00950 LF       EQU      $-OFFSET
A14F CDD105      00960          CALL     STATUS            ;IS PRINTER READY
A152 C24F76      00970          JP       NZ,LF
A155 3E0A        00980          LD       A,0AH             ;SET OUTPUT CHAR = <LF>
A157 32E837      00990          LD       (PRTR),A          ;OUTPUT IT TO PRINTER
A15A 212940      01000          LD       HL,LINCNT         ;POINT TO LINE/PAGE CNTR.
A15D 34          01010          INC      (HL)              ;INC LINE/PAGE CNTR
A15E 7E          01020          LD       A,(HL)            ;GET NEW VALUE OF CNTR
A15F FE36        01030          CP       54D               ;HAVE 54 LINES PRINTED ?
A161 C28176      01040          JP       NZ,EXIT
                 01050
                 01060 ;>>>>>>>>>>     OUTPUT  <FF>     <<<<<<<<<<
                 01070
7664             01080 FF       EQU      $-OFFSET
A164 CDD105      01090          CALL     STATUS            ;CHECK PRINTER STATUS
A167 C26476      01100          JP       NZ,FF
A16A 3E0C        01110          LD       A,0CH             ;SET OUTPUT CHAR = <FF>
A16C 32E837      01120          LD       (PRTR),A          ;PRINT IT
A16F 3600        01130          LD       (HL),00H          ;RESET LINE/PAGE CNTR
A171 23          01140          INC      HL                ;POINT TO PAGE CNTR
```

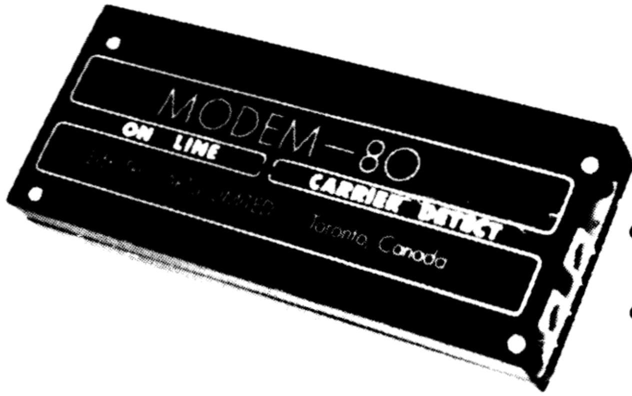
Program continues

MODEM·80

TRS-80 MODEL III VERSION AVAILABLE SHORTLY

INQUIRIES WELCOME

THE TRS-80* DATA COMMUNICATIONS SYSTEM FOR THE EIGHTIES



- Direct connects to your home telephone jack
- Installs in a minute
- Program swapping mode plus terminal mode
- All software included
- No RS232C interface is required
- Lowest price ever

The MODEM-80 and your TRS-80 Model I Level 2 disk or tape system make the perfect combination to maximize your data communications capabilities.

This direct connect, auto dialing/answering modem will transmit programs and data between MODEM-80 equipped TRS-80 systems. In addition, it will transform your computer into a full-function intelligent terminal. And, no RS-232C interface is required, so tape-based systems do not need an expansion interface. The result is the lowest priced data communications system of its type available anywhere.

Program-swapping with your friends is a snap with the MODEM-80. Both machine-language and BASIC programs can be easily exchanged over the telephone line. And, accessing your favorite computer club bulletin board or any other Bell 103-type compatible time-sharing facility is no problem either.

The TRS-80 to TRS-80 software, included with the MODEM-80, extends your system's BASIC language with 8 new communications-related functions. These can be easily incorporated into your own BASIC programs allowing, for example, "computer vs. computer" competitions. BASIC programs are also provided which support disk, tape and memory transfer of programs and data, including automatic line error detection and re-transmission. This approach combines function with flexibility, reliability and ease-of-use.

When operating in the intelligent terminal mode, you have over 35 functions from which to choose. These include 8 user-definable function keys, baud rate selection from 25 to 300, forward and backward buffer scrolling, parity control, full and half-duplex operation, disk and tape file load and save, automatic dialing, parallel printer support...the list goes on and on.

You can install a MODEM-80 on your system in one minute. The ribbon connector plugs into the back of your keyboard or expansion interface. Power is provided by your computer's power module so no additional 110 V outlets are needed. An extension cable with standard miniature phone plugs is included for direct connection to the telephone line and there is even a spare jack to plug in a phone, if desired. Since the TRS-80 bus is extended through the MODEM-80, you can leave the unit permanently connected and still plug in other devices as you expand your system.

The MODEM-80 complete with all software and User Guides is **only** U.S. **$219.00** plus** U.S. **$5.00** for shipping, handling and insurance. We accept money orders, VISA and MasterCard. The MODEM-80 hardware has a 90 day manufacturer's warranty.

INTRODUCING TOMORROW'S PRODUCTS—TODAY!

**Subject to Canadian rate of exchange at time of purchase

*TRS-80 is a trademark of Tandy Corp.

TO ORDER CALL **416-923-7961 ANYTIME**
AND ASK FOR ICROM'S ORDER DESK
OR WRITE
ICROM ENTERPRISES LTD. 40
1240 BAY ST.—SUITE 505
TORONTO, CANADA M5R 2A7

See List of Advertisers on page 386

Program continued

```
A172 34        01150         INC     (HL)           ;INC PAGE #
A173 CD8F76    01160         CALL    PTPAGE         ;PRINT PAGE HEADER
A176 0E0D      01170         LD      C,0DH          ;SET OUTPUT CHAR = <CR>
A178 CD3A76    01180         CALL    START          ;OUTPUT 3 <CR>'S TO
A17B CD3A76    01190         CALL    START          ;SEPERATE HEADER FROM
A17E CD3A76    01200         CALL    START          ;TEXT
7681           01210 EXIT    EQU     $-OFFSET
A181 79        01220         LD      A,C            ;RESTORE CHAR
A182 C9        01230         RET
               01240
               01250 ;******  COMMAND  <SHIFT> @  FOR EJECT ROUTINE  *******
               01260
               01270 ;       THIS ROUTINE USES A  "<SHIFT> `" TO REPLACE THE B
               01280 ; COMMAND. THE "<SHIFT> @" COMMAND WILL GIVE A FORM FEED
               01290 ; AND RESET THE LINE/PAGE AND PAGE # COUNTERS
               01300
7683           01310 EJECT   EQU     $-OFFSET
A183 212A40    01320         LD      HL,PAGCNT      ;POINT TO LINE/PAGE CNTR
A186 3600      01330         LD      (HL),00H       ;  "<SHIFT> @" SENT SO
A188 2B        01340         DEC     HL             ;     RESET PAGE COUNTER
A189 CD6476    01350         CALL    FF
A18C C3DA58    01360         JP      WARMST         ;EXIT TO WARM RESTART
               01370
               01380 ;********  PRINT PAGE HEADER ROUTINE **********
               01390
               01400 ;       THIS ROUTINE WILL PRINT THE PAGE # INFO
               01410 ; ON THE PAGE.
               01420
768F           01430 PTPAGE  EQU     $-OFFSET
A18F 218174    01440         LD      HL,MSG1        ;PT TO BEG OF PAGE HEADER
A192 CDAA76    01450         CALL    OUTSTR         ;OUTPUT MESSAGE
               01460
               01470 ;>>>>>>  HEX TO ASCII CONVERSION FOR PAGE #  <<<<<<<<
               01480
               01490 ;       THIS SECTION MAKES USE OF THE HEX TO ASCII
               01500 ; CONVERSION IN ROM. THE 16 BIT HEX VALUE TO BE CONVERTED
               01510 ; MUST BE STORED AT 4121H. AFTER THE CONVERSION THE "HL"
               01520 ; REGISTER CONTAINS THE ADDRESS OF THE MOST SIGINIFICANT
               01530 ; DIGIT OF THE NUMBER.
               01540
A195 3A2A40    01550         LD      A,(PAGCNT)     ;GET HEX PAGE #
A198 6F        01560         LD      L,A
A199 2600      01570         LD      H,00H          ;STORE PAGE COUNT INTO
A19B 222141    01580         LD      (4121H),HL     ;  CELL TO BE CONVERTED
A19E 3E02      01590         LD      A,2            ;LOAD # TYPE FLG FOR INT.
A1A0 32AF40    01600         LD      (40AFH),A      ; TO HEX INPUT
A1A3 CDBD0F    01610         CALL    0FBDH          ;  CONVERT, RESULTS  'HL'
A1A6 CDAA76    01620         CALL    OUTSTR         ;HL PR TO STRING. OUTPUT
               01630                                ;    PAGE #.
A1A9 C9        01640         RET
               01650
               01660 ;********    OUTSTR    ********
               01670
               01680 ;       THIS SECTION OUTPUTS A STRING FROM MEMORY TO THE
               01690 ; PRINTER. THE "HL" REGISTER MUST POINT TO THE FIRST
               01700 ; CHARACTER TO BE PRINTED. A 00H MUST BE THE LAST
               01710 ; CHARACTER.
               01720
76AA           01730 OUTSTR EQU      $-OFFSET
A1AA 4E        01740         LD      C,(HL)         ;GET CHAR TO BE PRINTED
A1AB 79        01750         LD      A,C            ;SET UP TO BE PRINTED
A1AC FE00      01760         CP      00H            ;END OF STRING ?
A1AE C8        01770         RET     Z              ; IF YES, RETURN
A1AF E5        01780         PUSH    HL
A1B0 CD8D05    01790         CALL    PRINT          ;OUTPUT CHAR.
A1B3 E1        01800         POP     HL
A1B4 23        01810         INC     HL             ;PT TO NEXT CHAR
A1B5 C3AA76    01820         JP      OUTSTR
               01830
A1B8 00        01840 DONE    DEFB    00H            ;THIS IS THE END OF THE
               01850                                ;PRINTER PRE-DRIVER.
               01860
               01870 ;*******************************************************
               01880 ;*         BLOCK MOVE AND INITIALIZATION              *
               01890 ;*******************************************************
               01900
               01910 ;       THIS SECTION IS WHERE THE ACTUAL EXECUTION
               01920 ; BEGINS. THIS SECTION WILL BLOCK MOVE THE EDTASM WITH
               01930 ; THE PRINTER PRE-DRIVER DOWN TO ITS ACTUAL EXECUTABLE
               01940 ; LOCATION OF 5500H.
               01950
               01960 ;>>>>>>>>>>  RELOCATE  EDTASM  <<<<<<<<<<
               01970
A1B9 F3        01980 ENTRY   DI
A1BA 210080    01990         LD      HL,8000H       ;BEG. OF RELOC. EDTASM
A1BD 110055    02000         LD      DE,5500H       ;MOVE EDTASM TO
A1C0 01B821    02010         LD      BC,DONE-ASMST  ;# OF BYTES TO BE MOVED
A1C3 EDB0      02020         LDIR                   ;MOVE IT !!
               02030
               02040 ;>>>>>>>>>  GET DATE & INSERT IN HEADER MSG  <<<<<<<<<<
               02050
A1C5 210BA2    02060         LD      HL,DATE        ;POINT TO SPACE AFTER
               02070                                ;  "DATE:" IN MSG2
```

Program continues

MODEL III FOR $175.00

Yes, double density is here! Two 40 track drives give you 368K of storage. And if thats not enough, four 80 track drives will give you 1.4 megabytes of on line storage. All this with the added speed of double density operation, not to mention 80% more storage per diskette.

No loss of your software library or conversion of your single density diskettes is necessary. DOSPLUS 3.3D reacts to the diskette. It will read single or double density with equal ease. It is not JUST a double density operating system, it is a double AND single density system. It is the first of it's kind. No one can offer you what we can! To change a disk to double density you need only format a double density disk, and then ONE command (TRANSFER), will bring over all your files. Very simple to use, but not necessary, because DOSPLUS will read them as they are now. Just insert single density diskettes and run (with our operating system). The machine will know what you are doing without having to be told ANYTHING!

If you want your computer to **chain functions,** then you need DOSPLUS 3.3D (for example, from powerup you could have your machine call another computer, scan a data base for appointments, recording any that appear on your printer; and then load in our new BASIC, setting files and mem/size, and start your program so your secretary can go right to work). With our BUILD, DO and AUTO, all of the preceding would be child's play. By the way, you could have also set the time and date, looked at the directory of all your drives as well as checked the free space map (which tells you not only how much space you have on a diskette, but also where it is and what it is used for) DOSPLUS is a truly intellegent, easy to use operating system that gives you all your computer is capable of delivering.

You probably guessed that for $175.00 what we are doing is turning your Model I into a Model III. It's so easy **anyone can do it in 10 minutes!** Why spend $2,500.00 for a Model III when you can get the same computing power, with our kit, and NO SOFTWARE CONVERSION, for only $175.00. Don't throw away your Model I, let us expand it! Move up to the world of double density.

You will reap the benefits of our **error-free software.** No miracles, just plain hard work and a lot of testing by experts and novices alike. Test us and judge yourself. We guarantee you will be 100% satisfied. If you are the first to find a legitimate "BUG" in our software you will be rewarded with a brand new $100 dollar bill.

Your systems disk will come complete with an all new single/double density disk editor called DISKZAP, and a BASIC program compressor called CRUNCH. Also included is PURGE, a utility to make the mass removal of unwanted files from a diskette easy, and RESTORE, a program that makes recovering a dead file as easy as typing in a command line. TRANSFER is just as it sounds, a program that moves all files (except systems) from one diskette to another. Single density to double or vice versa. CLEARFILE is used to zero data files on a diskette for a "clean slate". DISKDUMP is a new machine language sector display/modify program that works with filespecs instead of tracks and sectors. Used in conjunction with DISKZAP, you will have more disk editing power than ever before, with less frustration than was ever thought possible.

You will now be able to use all your **DOS commands from our new BASIC** with the CMD feature. And how about variable length records that really work, first time, EVERY time! This will allow you to use the ISAM programming technique for vastly improved handling of large data bases and lightning speed unheard of in BASIC. (ISAM stands for Indexed Sequential Access Method).

You can create programs that will run on MOD III DOSPLUS, yes, 100% compatability.

If inflated computing power without an inflated price tag is what you're after, contact us at the address below.

MICRO SYSTEMS SOFTWARE INC ✓384

Specializing in the Tandy Line

(305) 983-3390

5846 Funston Street
Hollywood, FL 33023

CALL TOLL FREE FOR FAST SERVICE
(800) 824-7888, OPERATOR 193
FOR VISA/MASTERCHARGE/C.O.D. ORDERS
California dial (800) 852-7777, Operator 193
Alaska and Hawaii dial (800) 824-7919, Operator 193
TOLL FREE LINES WILL ACCEPT ORDERS ONLY!
For Applications and Technical information, call (305) 983-3390 or drop us a card

Dealers inquiries invited

✓See List of Advertisers on page 386

Program continues

```
A1C8 CD7044   02080         CALL   4470H            ;CONVERT DATE & STORE IN
              02090                                 ;     MSG2
              02100
              02110 ;>>>>>>  MOVE HEADER MSG OVER APPARATE SIGN ON MSG  <<<<<
              02120
A1CB 218074   02130         LD     HL,APPMSG        ;POINT TO APPARAT SIGN ON
A1CE 3EA0     02140         LD     A,0A0H           ;SET 1ST CHAR IN SIGN ON
A1D0 77       02150         LD     (HL),A           ; MSG TO MST TERMINATOR
A1D1 23       02160         INC    HL               ;POINT TO NEXT CELL
A1D2 1100A2   02170         LD     DE,MSG2
A1D5 EB       02180         EX     DE,HL
A1D6 015000   02190         LD     BC,80D           ;BYTE COUNTER
A1D9 EDB0     02200         LDIR                    ; MOVE MSG2 OVER APP. MSG
              02210
              02220 ;>>>>>>  REPLACE " B " WITH " <SHIFT> @ " <<<<<<<<
              02230
A1DB 212F5B   02240         LD     HL,862FH-OFFSET ;ASM'S " B " COMMAND
A1DE 3E60     02250         LD     A,60H            ;" <SHIFT> @ "
A1E0 77       02260         LD     (HL),A           ; " B " --> " <SHIFT> @ "
A1E1 23       02270         INC    HL               ;PT LSB OF ROUTINE ADD
A1E2 118376   02280         LD     DE,EJECT         ;LSB OF "<SHIFT> @" ADD
A1E5 73       02290         LD     (HL),E           ;CHANGE LSB OF CMD ADD
A1E6 23       02300         INC    HL               ;PT TO MSB
A1E7 72       02310         LD     (HL),D           ;CHANGE MSB
              02320
              02330 ;>>>>>>>  LINK PRE-DRIVER TO ROM PRINTER DRIVER  <<<<<<<<
              02340
A1E8 213A76   02350         LD     HL,763AH         ;"HL" = ADD OF PRE-DRIVER
A1EB 222640   02360         LD     (4026H),HL       ;PATCH IN PRINTER DCB
              02370 ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
              02380
              02390 ;$   NOTE :  THIS SETS HEATH H-14 TO 96 CHAR / LINE
              02400 ;$ DELETE OR CHANGE TO FIT YOUR PRINTER.                  $
              02410
A1EE 0E16     02420         LD     C,16H            ; OUTPUT CHAR =  <ESC>
A1F0 CD4576   02430         CALL   OUTPUT           ;OUTPUT IT
A1F3 0E75     02440         LD     C,75H            ; OUTPUT CHAR = <LC U>
A1F5 CD4576   02450         CALL   OUTPUT
A1F8 0E18     02460         LD     C,18H            ; OUTPUT CHAR = <CTL X>
A1FA CD4576   02470         CALL   OUTPUT
              02480
              02490 ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
              02500
A1FD C3006F   02510         JP     6F00H            ;START EXECUTION OF ASM2
              02520
              02530 ;>>>>>>>>>>  HEADER MESSAGE  <<<<<<<<<
              02540
A200 20       02550 MSG2    DEFM   '      DATE: '
A20B 20       02552 DATE    DEFM   '         '        ;DATE INSERTED HERE
A213 20       02554         DEFM   '      NEWDOS PLUS'
A223 20       02560         DEFM   '        Z-80 ASSEMBLER'
A238 20       02570         DEFM   '        *****  PAGE '
A24B 00       02580         DEFB   00H
A1B9          02590         END    ENTRY
00000 TOTAL ERRORS
```

Program Listing 3

```
              00100 ;***********************************************************
              00110 ;*                                                         *
              00120 ;*               OPTIONAL  'FORM FEED' ROUTINE             *
              00130 ;*                                                         *
              00140 ;***********************************************************
              00150
              00160
              00170 ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
              00180 ;$                                                         $
              00190 ;$        NOTES:
              00200 ;$        1)   THIS CAN REPLACE THE SECTION OF THE         $
              00210 ;$          FORM FEED ROUTINE THAT OUTPUTS THE 0CH.        $
              00220 ;$        2)   THE REFERENCE COUNTER (IX+03) IS 67         $
              00230 ;$         LINES / PAGE. THIS CAN BE CHANGED BY            $
              00240 ;$        POKING THE VALUE DESIRED, OR BY ADDING:          $
              00250 ;$        LD       HL,4028H                                $
              00260 ;$        LD       (HL),XX                                 $
              00270 ;$        WHERE XX = THE # OF LINES/PAGE DESIRED           $
              00280 ;$        INTO THE INITIALIZATION ROUTINE                  $
              00290 ;$                                                         $
              00300 ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
              00310
              00320
0000 DDE5     00330         PUSH   IX
0002 CD9F05   00340         CALL   059FH            ;FINE # LINES LEFT ON
              00350                                 ;PAGE, AND OUTPUT AS
              00360                                 ;<LF>'S.
0005 DD3405   00370         INC    (IX+05)          ;INC PAGE COUNTER
0008 DDE1     00380         POP    IX               ;INC PAGE COUNTER
              00390
0005          00400         END    ;  ***  DELETE THIS LINE  ***
00000 TOTAL ERRORS
```

dress of the first character to be printed, and that the last character be a 00H. OUTSTR gets the character pointed to by the HL register and compares it to 00H. If it is not a 00H, the ROM printer driver is called to output the character. When a 00H character is encountered, a return from the subroutine is executed without printing that character.

The page number is stored as a hexadecimal value at address 402AH, which is an unused cell in the printer DCB. The ROM hex to ASCII conversion routine is used to convert the page number to a printable value. It is assumed a program listing is less than 255 pages, so only one byte is needed for the counter. The HL register is loaded with the hex value to be converted which is then stored at 4121H. The hex to ASCII routine converts the value stored at 4121H and stores the results somewhere in memory. However, the HL register contains the address of the most significant digit converted. After the conversion, OUTSTR is again called to output the page number.

Another feature included with the page headers is incorporation of the date. When the DOS is booted, enter the date into the system by using the Date command. Then when EDTASM is loaded, it will convert the date to ASCII and store it in the header message. This is done in the initialization section.

Finally, we need some way to reset the page counter and possibly produce a top-of-form. To accomplish this, the B command (return to Basic) is replaced in the command lookup table by an @. Then the reset address of 0000H is changed to the start of the form feed routine. Again, this is accomplished during initialization.

Initialization

The initialization section has many functions. One important one is its ability to move EDTASM down to 5500H where Apparat intended it to run. This is accomplished by lines 1960–2020. Next, the date is calculated and stored in the page header message area of memory by lines 2060 and 2080.

Replacing Apparat's sign-on message is handled by lines *2110-2200.* Apparat uses A0H as their end of string terminator, so the HL register is loaded with the starting address of their message. The accumulator is loaded with A0H and then stored at the address pointed to by the HL register. This pointer is then bumped, and the DE register is loaded with the starting address of the page header message. HL and DE are exchanged, to conform to the requirements of the LDIR instruction, and BC is loaded with the number of bytes to be moved. The LDIR instruction actually moves the message into the old sign-on area. Finally, the B command and its associated jump address are replaced with the @ sign and the formfeed routine's address, and the printer driver's address in the DCB is replaced.

Labels

Many labels in the program have an offset of 2B00H because EDTASM was moved from 5500 to 8000H to enable us to work on it, save EDTASM and the patches. If it were left at 5500H, any return to DOS would have overlaid another program. The program must be moved up to 8000H by using the LMOFFSET program in NEWDOS. It will attach a short routine to the end, like the one in lines 1810-1850, followed by the jump in line 2210. The coding between lines 1850 and 2210 is not required for the printer pre-driver and will then be written over by EDTASM after it is executed.

Once the program has been written and saved to disk, you must merge the two programs. Use LMOFFSET to move EDTASM to 8000H, and resave it to disk. Then return to DOS. Use Load to put program into memory (*do not* execute either program). Next load Tapedisk into memory, and save the programs from 8000H to A260H, with the entry point at Entry or A1B9H. I named the modified version of EDTASM ASM2 to avoid confusion as to which EDTASM is loaded.■

An empire in search of lost greatness.

Star Colony

John Beringer
2729 West Sahara #2
Las Vegas, NV 89102

The morning is bright and clear. A few clouds dot the sky at the higher altitudes. It is a perfect day for a launch.

To commemorate this great occasion, the Corps band has been borrowed from the World Air Force. The band's music, harsh and tinny in your ears, blares across the allo-concrete. Your chiefs and aides stand stiffly erect, impatient with all this pomp and circumstance.

At last the music ends. There is the usual round of hand-shaking and well-wishing. When the farewells are done, you and your men board the shuttle. The craft launches and swiftly climbs to space where it mates with your enormous starship, the first to be built in a long time since the Great War, Interstellar War I, the war against the D'nim (see *Star Guard* in an earlier issue).

It is only reasonable that humanity should make the long crawl back into space, you reflect during docking procedures. What is tragic is that it took so long for the Earth to recover from the ravages of IWI. Triumphant, yet with all her colonies lost, Earth has had a hard time pulling herself up to her former level. Your ship (you have the privilege of naming it) is a gigantic vessel, built solely for exploration and recolonization. You are to guide her in reopening the space lanes and reestablishing the footholds Earth once had.

Object of the Game

The object of this game is to colonize and explore as many star systems as the player can in the time alloted to him. (The time is randomly determined at the start of the game.) Upon departure, you have 100 per cent operation power and a full stock of supplies, along with 5000 prospective colonists in stasis. As power and supplies diminish, it will become necessary to replenish them. There are two methods of doing this. You can mine explored systems for radioactive fuel elements and raw materials for supplies, which takes away from mission time (with the poorer systems yielding less and requiring more mining time than the richer systems). A decision may be made as to whether or not the player should resupply completely. Also, fuel and supplies will be used up during the years that mining and refining are taking place.

Another way to restock the ship is to revisit an established colony. If the colony is doing well, the members will gladly give you the supplies you need. A word of warning—if you take too much from the colony, you could tax the colony severely and cause it to fail. Obviously, the colonies to revisit are those which have high-yield concentrations of raw materials.

To establish a colony in a newly explored star system, the proper number of colonists (decided upon by the Captain) are brought out of stasis and sent down to the most hospitable planet available. They are given supplies, and then left on their own. A return visit will show how the colony is faring. Any colony that has a population of 500 or more members is considered thriving and able to aid in resupplying the ship. Some star systems are so poor

in materials that they should be bypassed and not even considered for colonization. Setting down colonists on these systems will be useless since they have almost no chance to survive and grow.

If you run out of time, the game ends. If you run out of fuel or supplies, your crew may toss you out the air lock, as there is no rescue available from Earth. If you last to the end of your mission, your performance will be evaluated and your superiors on Earth will decide how to reward you (or punish you!).

How To Play

There are four general commands available to you: Map, Scan, Jump, and Status. Different options are available within each command.

Map displays all the star systems within the range of your starship. Any attempt to explore outside the designated area will result in the starship becoming lost or even destroyed in some cases. An option of this command is the History subcommand. If selected, all explored stars will be shown as an E and all colonized systems will be shown as a C. The starship is shown as an up arrow (↑). This display allows you to see which stars you have visited and colonized.

A second option of the Map command is called the Rangefinder. With this option, you can pinpoint a particular star on the map. By using the keyboard arrows, you move the rangefinder, "+", onto a selected star. Then the spatial co-ordinates (x, y, and z) may be displayed, as well as the star's distance from your starship.

Scan enables you to explore a star system, discovering such information as the number of planets in the system, the number of habitable (Terra-type) planets, the grade of raw materials available and floral/faunal evolutionary levels rang-

Program Listing. Star Colony

```
2 CLEAR200
100 DATA 0,0,0,480,-1,-1,-4,541,-4,0,-8,468,-4,-1,16,532,-5,0,-1
3,465,-5,2,5,337,-6,2,1,334
102 DATA -6,-1,-19,526,-1,-6,14,861,0,-6,0,864,1,-6,10,867,2,-5,
17,806,4,-3,-10,684
104 DATA 5,-3,11,687,6,-1,7,562,3,0,-21,489,7,1,8,437,9,2,15,379
,6,3,-2,306,-3,5,-18,151
106 DATA -7,4,11,203,-9,1,0,389,3,-7,-18,937,5,-6,7,879,6,-7,-8,
946,7,-7,-12,949,9,-4,-3,763
108 DATA 7,-2,-7,629,7,5,-3,181,6,7,-2,50,-2,7,-2,26,8,5,8,184
110 DATA SOL, ALPHA CENTURI, INNES STAR, AC 79, CC 658, LALANDE
21185, WOLF 359, LUYTEN 68-28, BD +68, BARNARD'S STAR, BD +59, S
IGMA DRACONIS, EPSILON INDI, KRUEGER 60, ROSS 248, BETA HYDRI, G
ROOMBRIDGE 34, ETA CASSIOPEIAE, LUYTEN 726-8
112 DATA LUYTEN 97-12, BD +50, ROSS 128, DELTA PAVONIS, 61 CYGNI
, CD -39, CD -49, LUYTEN 789-6, CD -36, TAU CETI, EPSILON ERIDAN
I, SIRIUS, RHO ERIDANI
910 DIMST(32,5):DIMST$(32,2):DIMPL(32,3)
915 SP$="                                                 "
1000 REM
1010 CLS
1020 INPUT"LAST NAME";NA$:IFNA$=""THENPRINT"NEED A NAME.":GOTO10
20ELSEIFLEN(NA$)>15THENPRINT"15 CHARACTERS OR LESS, PLEASE.":GOT
O1020
1030 PRINT:INPUT"YOUR SHIP'S NAME";SH$:IFSH$=""THENPRINT"NEED A
NAME FOR YOUR SHIP.":GOTO1030ELSEIFLEN(SH$)>15THENPRINT"15 CHARA
CTERS OR LESS, PLEASE.":GOTO1030
1040 PRINT:PRINT"WELCOME ABOARD THE ";SH$;", CAPTAIN ";NA$;"!"
1050 YE=RND(25)+35:PRINT:PRINT"YOU HAVE";YEARS;"YEARS BEFORE YOU
 MUST RETURN TO EARTH.":PRINT:PRINT"GOOD LUCK!"
1070 RESTORE:FORA=1TO32:FORB=1TO4:READST(A,B):NEXTB:ST(A,5)=0:NE
XTA
1075 FORA=1TO32:READST$(A,1):ST$(A,2)=".":NEXT
1080 PW=100:CL=5000:SU=100:CR=4:SC$="Y":EN$="Y":LA$="Y":SR=0:X=0
:Y=0:Z=0:AD=480:OV=2:IN=1:VI=0:ED=0:SD=0:LD=0:CD=2310:YR=0:M3$="
":M4$="":M5$="":M6$="":M7$="":C9$=CHR$(28)
1085 PRINT"PRESS ANY KEY TO BEGIN."
1090 IFINKEY$=""GOTO1090
1100 CLS
1120 PRINT@960,CHR$(30);:INPUT"MAP, JUMP, SCAN, OR STATUS";R$:IF
R$="MAP" GOTO2000ELSEIFR$="JUMP" GOTO3000
        ELSEIFR$="SCAN" GOTO4000ELSEIFR$="STATUS" GOTO5000ELSEGO
TO1120
2000 IFMA=1THENMA=0:GOTO2100
2010 CLS:FORN=1TO32:PRINT@ST(N,4),".";:NEXTN:PRINT@AD,"[";:PRINT
@960,"WANT STATUS OF VISITED STAR SYSTEMS? Y/N";:GOSUB9050:IFR=0
GOTO2100
2015 PRINT@415,"SOL";:PRINT@768,"[ - SHIP";:PRINT@832,"E - EXPLO
RED ONLY";:PRINT@896,"C - COLONIZED";
2020 FORN=1TO32:IFST$(N,2)="." NEXTELSEPRINT@ST(N,4),ST$(N,2);:N
EXT
2025 PRINT@AD,"[";
2100 IFPEEK(14338)=64GOTO2100ELSEPRINT@960,CHR$(30);:PRINT@960,"
WANT RANGEFINDER? Y/N";:GOSUB9050:IFR=0GOTO8000ELSEPRINT@960,CHR
$(30);:PRINT@960,"KEYBOARD ARROWS MOVE RANGEFINDER  +.  KEY I FO
R INFORMATION.";
2105 NA=AD
2110 LA=NA:LS=PEEK(NA+15360):PRINT@NA,"+";
2120 IFPEEK(14338)=64THENGOTO8000ELSEIFPEEK(14338)=2GOTO2600ELSE
W=PEEK(14400):IFW=8GOTO2200ELSEIFW=16GOTO2300ELSEIFW=32GOTO2400E
LSEIFW=64GOTO2500ELSEGOTO2120
2200 IFNA-64<0PRINT@960,CHR$(30);:GOSUB9000:GOTO2120
2210 NA=NA-64
2220 POKELA+15360,LS:GOTO2110
2300 IFNA+64>959GOSUB9000:GOTO2120ELSENA=NA+64:GOTO2220
2400 IFNA-1<0GOSUB9000:GOTO2120ELSENA=NA-1:GOTO2220
2500 IFNA+1>959GOSUB9000:GOTO2120ELSENA=NA+1:GOTO2220
2600 FORN=1TO32:IFST(N,4)=NAGOTO2650ELSENEXT
2610 GOSUB9010:PRINT@768,"NO STAR THIS LOCATION.";:M5$="":M6$=""
:M7$="":GOTO2120
2650 GOSUB9010:M5$=ST$(N,1) + " SYSTEM":XX=ST(N,1):YY=ST(N,2):ZZ
=ST(N,3):GOSUB2653:GOTO2670
2653 IFSGN(X)=SGN(XX)THENDX=X-XXELSEDX=ABS(X)+ABS(XX)
2654 IFSGN(Y)=SGN(YY)THENDY=Y-YYELSEDY=ABS(Y)+ABS(YY)
2655 IFSGN(Z)=SGN(ZZ)THENDZ=Z-ZZELSEDZ=ABS(Z)+ABS(ZZ)
2660 DI=SQR(DX[2+DY[2+DZ[2):RETURN
2670 M6$="X, Y, Z: " + STR$(ST(N,1)) + ", " + STR$(ST(N,2)) + ",
 " + STR$(ST(N,3)):M7$="DISTANCE- " + STR$(DI) + " LY.":PRINT@76
8,M5$;:PRINT@832,M6$;:PRINT@896,M7$;:PRINT@960,CHR$(30);:PRINT@9
60,"KEY 'N' IF DONE ELSE MOVE RANGEFINDER.";:GOTO 2120
3000 IFPW<8THENPRINT"NOT ENOUGH POWER LEFT TO JUMP AGAIN, CAPTAI
N.":GOTO8000ELSEIFC9$=CHR$(10)THENGOTO3010ELSECLS
3001 IFM5$><"" PRINTM5$:PRINTM6$:PRINTM7$
3005 GOSUB3010:GOTO3025
3010 PRINT"CURRENT LOCATION OF THE ";SH$;" - ";:FORN=1TO32:IFAD=
ST(N,4)ANDZ=ST(N,3)THENPRINTST$(N,1);" SYSTEM"ELSENEXTN:PRINT"IN
 DEEP SPACE"
3020 PRINT" X"," Y"," Z":PRINTX,Y,Z:IFEN$="Y" THENMX=7.5
3021 RETURN
3025 PRINT"WANT TO JUMP? Y/N":GOSUB 9050:IFR=0GOTO3999
3030 INPUT"DESTINATION X CO-ORDINATE";XD:DD=XD:GOSUB9500:IFR=0GO
TO3030
3033 INPUT"DESTINATION Y CO-ORDINATE";YD:DD=YD:GOSUB9500:IFR=0GO
TO3033
3036 INPUT"DESTINATION Z CO-ORDINATE";ZD:DD=ZD:GOSUB9500:IFR=0GO
TO3036
3038 XX=XD:YY=YD:ZZ=ZD:GOSUB2653
3040 IFDI<=MXGOTO3060ELSEPRINT"DISTANCE TO JUMP IS";DI;"LIGHTYEA
RS.":IFMX<7.5THENPRINT"OUR ENGINES ARE DAMAGED, CAPTAIN, AND WE
CAN ONLY JUMP";MX:PRINT"LIGHTYEARS.  ENTER EXACT CO-ORDINATES TO
 JUMP.":GOTO3025
3041 PRINT"THIS IS";DI-MX;"LY GTR THAN THE MAXIMUM SAFE DISTANCE
":PRINT"SHALL WE RISK OVERJUMP, CAPTAIN?  Y/N":GOSUB9050:IFR=1GO
TO3050
3042 PRINT"SHALL WE SET COURSE AND JUMP AS FAR AS WE CAN SAFELY?
 Y/N":GOSUB9050:IFR=0GOTO3025ELSEDX=0:DY=0:DZ=0:GOSUB9300
3043 IFX-XD<0THENDX=DX+IN:X=X+INELSEIFX-XD>0THENDX=DX-IN:X=X-IN
3044 GOSUB9250:IFR=1GOTO3048
3045 IFY-YD<0THENDY=DY+IN:Y=Y+INELSEIFY-YD>0THENDY=DY-IN:Y=Y-IN
3046 GOSUB9250:IFR=1GOTO3048
3047 IFZ-ZD<0THENDZ=DZ+IN:Z=Z+INELSEIFZ-ZD>0THENDZ=DZ-IN:Z=Z-IN
3048 GOSUB9250:IFR=0GOTO3043ELSEGOTO3070
3050 GOSUB9300:OV=DI-7.5:IFOV>10THENRI=2ELSEIFOV>5THENRI=3ELSEIF
OV>2THENRI=4ELSEIFOV>0THENRI=5
3052 OV=RND(RI):IFOV>1THENPRINT:PRINT"WE MADE IT!":PRINT:X=XD:Y=
YD:Z=ZD:GOTO3070ELSEFORN=1TO200:NEXTN:PRINT"CAPTAIN!  WE HAVE OV
ERJUMPED!"
3053 X=RND(22):Y=RND(18)-1:Z=RND(41)-21:IFX<24ANDY>11THENGOTO305
3
3054 IFX<22ANDY<15THENGOTO3056ELSEPRINT:PRINT:PRINT"SCANNERS IND

ICATE SPATIAL TIDAL FORCES INCREASING BEYOND OUR    HULL'S ENDUR
ANCE.  wE ARE BEING RIPPED APART!":FORN=1TO1500:NEXTN:GOSUB9200:
CLS:FORN=1TO300:NEXTN
3055 PRINT"SO MUCH FOR THE ";SHIPNA$:PRINT"AND HER CAPTAIN!":PRI
NT:END
3056 PRINT:PRINT"WE'RE LOST, CAPTAIN!":PRINT:X=X-11:Y=Y-7:GOTO30
70
3060 IFX=XDANDY=YDANDZ=ZDTHENPRINT"NO JUMP WAS MADE - ZERO DISTA
NCE.":GOTO3025
3063 PRINT"DISTANCE IS";DI;"LY - A SAFE JUMP. SHALL WE JUMP? Y/N
":GOSUB9050:IFR=0GOTO3025ELSEX=XD:Y=YD:Z=ZD:GOSUB9300
3070 IFABS(X)>10ORABS(Y)>7ORABS(Z)>21THENPRINT:PRINT"CAPTAIN! WE
'VE STUMBLED INTO A BLACK HOLE!":PRINT"THE HOLE'S GRAVITY TIDES
ARE RIPPING US APART!":FORN=1TO1500:NEXTN:GOSUB9225:CLS:FORN=1TO
300:NEXTN:GOTO3055
3075 IFOV>1THENPRINT"JUMP WAS SUCCESSFUL."ELSEOV=2
3080 AD=(-Y+7)*64+(X+10)*3+2:VI=0:YR=YR+.6:PW=PW-8:C9$=CHR$(10):
IFOV>1THENGOSUB3010
3090 IFEN$="N" GOTO3999ELSEIFRND(15)=1THENEN$="N":EF=RND(90)/10+
1.2:MX=RND(4):PRINT"CAPTAIN!  THAT LAST JUMP DAMAGED OUR ENGINES
.  WE'LL ONLY BE    ABLE TO JUMP";MX;"LY!  THE ENGINEERS SAY IT'
LL TAKE";EF;"YEARS":PRINT"TO FIX THE ENGINES.":ED=CD+EF
3999 M5$="":GOTO8000
4000 IFPW<2THENPRINT"NOT ENOUGH POWER FOR LONG RANGE SCANNERS, C
APTAIN.":GOTO4900ELSEPW=PW-2
4001 YR=YR+.2:IFVI=1THENPRINT"YOU'VE ALREADY SCANNED THIS SYSTEM
, CAPTAIN.":GOTO4900ELSEVI=1:CLS:PRINT"SCANNING...":FORN=1TO32:I
FAD=ST(N,4)ANDZ=ST(N,3)THENGOTO4100ELSENEXTN:PRINT"NO SYSTEMS WI
THIN SCANNING RANGE.":GOTO4900
4100 IF N>1 GOTO 4200 ELSE PRINT"THE SOL SYSTEM":PRINT" X"," Y",
" Z":PRINTSTAR(N,1),STAR(N,2),STAR(N,3):PRINT"PLANETS: 9":PRINT"
TERRA-TYPES: 1":PRINT"MED GRADE METALS":PRINT"HIGH LEVEL FAUNAL/
FLORAL TYPES":PRINT:PRINT"WHAT ARE YOU DOING HERE, ";NA$;"?"
4105 PRINT"GET OUT THERE AND COLONIZE!!!":GOTO4900
4200 IFST(N,5)>0GOTO4300
4205 PRINT"THE ";ST$(N,1);" SYSTEM":PRINT" X"," Y"," Z":PRINTST(
N,1),ST(N,2),ST(N,3):P1=RND(15):PL(N,1)=P1:PRINT"PLANETS:";PL(N,
1):P2=RND(3)-1:IFP2>P1THENP2=0
4210 TY=RND(10):IFTY=10THENTY=99:P2=0ELSEIFP2<1THENTY=98
4215 PL(N,2)=P2:PRINT"TERRA-TYPES:";P2:
4220 ST(N,5)=TY:GOSUB9405:PRINTM3$:PRINTM4$:ST$(N,2)="E":PRINT
4221 IFTY=6ORTY=9THENNB=RND(4)ELSEGOTO4229
4222 IFNB>1GOTO4229ELSEST(N,5)=TY+10:PRINT:PRINT"AMBUSHED BY ALI
EN HOSTILES!":GOSUB9200
4224 PW=PW-RND(PW)/2:PRINT"POWER DRAINED TO";PW:PRINT"WE'D BETTE
R GET OUTTA HERE!":VI=0:PRINT:GOTO4900
4229 IFCL<1GOTO4250ELSEPRINT"SHALL WE COLONIZE THIS SYSTEM, CAPT
AIN ";NA$;"? Y/N":GOSUB9050:IFR=0GOTO4800
4230 IFTY=1ORTY=2ORTY=99ORTY=98THENPRINT"COLONISTS DON'T WANT TO
 GO, CAPTAIN!  SHALL WE FORCE THEM? Y/N":GOSUB9050ELSEGOTO4250
```

Program continues

Program continued

```
4240 IFR=0GOTO4800ELSETY=TY+10:ST(N,5)=TY
4250 IFCL<1PRINT"CAN'T COLONIZE - WE HAVE NO COLONISTS, CAPTAIN!
":GOTO4800ELSEINPUT"HOW MANY COLONISTS SHALL WE SET DOWN, CAPTAI
N";NB:IFNB>CLTHENPRINT"WE DON'T HAVE THAT MANY COLONISTS, SIR.":
GOTO4250
4251 IFNB><INT(NB)PRINT"CAN'T SEND PART OF A COLONIST, SIR!":GOT
O4250ELSEIFNB<1PRINT"CAN'T COLONIZE WITHOUT COLONISTS, SIR!":GOT
O4229
4260 CL=CL-NB:PL(N,3)=NB:NB=INT(NB/500):IFTY>0ANDTY<5THENMI=NB*5
:NB=2ELSEIFTY>4ANDTY<8THENMI=NB*2:NB=3ELSEIFTY=8ORTY=9THENMI=NB:

NB=4ELSEIFTY=99ORTY=98THENMI=50:NB=2
4265 PRINT"WHAT PERCENT OF OUR SUPPLIES SHALL WE GIVE THE COLONI
STS,":INPUT"SIR";PE:IFPE<0PRINT"CAN'T GIVE AWAY NEGATIVE SUPPLIE
S, SIR!":GOTO4265ELSEIFPE>SUTHENPRINT"WE DON'T HAVE THAT MUCH, C
APTAIN!":GOTO4265ELSESU=SU-PE:IFPE>=MI/50THENNB=NB+1
4266 R=RND(NB):IFR=1THENPL(N,3)=0
4280 STAR$(N,2)="C":GOTO 4800
4300 PRINT"I RECOGNIZE THIS SYSTEM, CAPTAIN!"
4302 PRINT"THIS IS THE ";ST$(N,1);" SYSTEM.":PRINT" X"," Y"," Z"
:PRINTX,Y,Z:PRINT"PLANETS:";PL(N,1):PRINT"TERRA-TYPES:";PL(N,2):
TY=ST(N,5):GOSUB 9405:PRINTM3$:PRINTM4$
4304 IFST$(N,2)="C" GOTO4400
4310 IFST(N,5)=16ORST(N,5)=19THENGOSUB9200:PRINT:PRINT"AMBUSHED
BY ALIEN HOSTILES AGAIN!":PRINT"DON't YOU EVER LEARN?":GOTO4224
4350 PRINT"WE EXPLORED THIS SYSTEM YEARS AGO";:IFCL<1GOTO4800ELS
EPRINT", BUT DIDN'T COLONIZE.  DO YOU WANT TO COLONIZE NOW? Y/N"

4360 GOSUB9050:IFR=1GOTO4230ELSEGOTO4800
4400 PRINT"WE COLONIZED THIS SYSTEM YEARS AGO.":IFPL(N,3)>0GOTO4
420
4405 PRINT"BUT OUR COLONY DIDN'T SURVIVE!":IFCL<1GOTO4800ELSEPRI
NT"SHALL WE TRY AGAIN? Y/N":GOTO4360
4420 IFST(N,5)<10ORST(N,5)>19THENGOTO4430ELSEIFRND(4)>1THENPRINT
"ATTACKED BY ANGRY COLONISTS!":GOSUB9200:PRINT"THEY REMEMBER BEI
NG FORCED ASHORE AGAINST THEIR WILL!":GOTO4224
4425 ST(N,5)=ST(N,5)-10:PRINT"THE COLONISTS FINALLY FORGIVE US F
OR FORCING THEM ASHORE!"
4430 PRINT"POPULATION STANDS AT";PL(N,3):IFPL(N,3)>499GOTO4450EL
SEPRINT"THE COLONY SEEMS TO BE SURVIVING, CAPTAIN, BUT CAN'T SPA
RE ANY  SUPPLIES. WE'LL HAVE TO MINE THOSE OURSELVES IF WE WANT
THEM.":NB=3:GOTO4455
4450 PRINT"THE COLONY HERE IS THRIVING, SIR.":PE=RND(100-PW):PRI
NT"THEY WANT TO GIVE US";PE;"% FUEL AND SUPPLIES.  SHALL WE TAKE
  IT? Y/N":GOSUB9050:IFR=0THENNB=4ELSENB=2:SU=SU+PE:PW=PW+PE:IFS
U>100THENSU=100
4455 GR=RND(NB):IFGR=1THENPL(N,3)=INT(PL(N,3)*.91)ELSEPL(N,3)=IN
T(PL(N,3)*1.21)
4490 GOTO4800
4800 REM
4810 IFTY>9THENTY=TY-10
4815 IFTY>0ANDTY<4THENTI=.4ELSEIFTY>3ANDTY<7THENTI=.2ELSEIFTY>6A
NDTY<10THENTI=.1ELSEIFTY=88ORTY=98THENTI=.6ELSEIFTY=89ORTY=99THE
NTI=5
4820 PRINT:PRINT"SIR, SHOULD WE MINE THIS SYSTEM FOR RADIOACTIVE
S (FOR FUEL)? OURPOWER LEVEL IS";PW;"%.  Y/N":GOSUB9050:IFR=0GOT
O4850
4825 INPUT"HOW MUCH SHOULD WE MINE? 0-100";PE:IFPE>100PRINT"WE C
AN ONLY CARRY 100%, SIR.":GOTO4825ELSEIFPE>0GOTO4827ELSEPRINT"NO
NE, SIR? Y/N":GOSUB9050:IFR=1GOTO4850ELSEGOTO4825
4827 IFINT(PW+PE)>100 THENPRINT"BUT SIR, WE ONLY NEED";100-PW;"%
":GOTO4825
4830 NBR=TI*PE:PRINT"IT WILL TAKE";NB;"YRS TO MINE WHAT YOU WANT
, CAPTAIN.":PRINT"SHALL i BEGIN THE MINING PROCEDURES? Y/N":GOSU
B9050:IFR=0GOTO4820
4835 YR=YR+NB:PW=PW+PE:PRINT:PRINT"MINING OPERATIONS COMPLETE":P
RINT"POWER LEVEL NOW STANDS AT";PW-NB;"%":PRINT
4850 IFTY>80PRINT:PRINT"INSUFFICIENT RAW MATERIAL OF THE TYPE WE
 NEED TO CONVERT TO     SUPPLIES, SIR.  WE'LL HAVE TO TRY ANOTHE
R SYSTEM.":PRINT:GOTO4900
4860 IFTY=1ORTY=4ORTY=7THENTI=.4ELSEIFTY=2ORTY=5ORTY=8THENTI=.2E
LSEIFTY=3ORTY=6ORTY=9THENTI=.1
4861 IFSU-YR*2<.1THENSP=0ELSESP=SU-YR*2
4862 PRINT:PRINT"SUPPLIES STAND AT";SP;"%, SIR.  SHOULD WE MANUF
ACTURE SOME":PRINT"BEFORE WE LEAVE THIS SYSTEM? Y/N":GOSUB9050:I
FR=0GOTO4900
4865 INPUT"HOW MUCH SHOULD WE MANUFACTURE, CAPTAIN";NB:IFNB>0GOT
O4870ELSEPRINT"NONE, SIR? Y/N":GOSUB9050:IFR=1GOTO4900ELSEGOTO48
65
4870 IFINT(SU-YR*2+NB)>100THENPRINT"BUT SIR, WE ONLY NEED";100-S
U+YR*2:GOTO 4865ELSEPE=NB*TI:PRINT"IT WILL TAKE";PE;"YRS TO CONV
ERT WHAT YOU WANT. SHALL i BEGIN  THE CONVERSION, SIR? Y/N":GOSU
B9050:IFR=0GOTO4865
4875 YR=YR+PE:SU=SU+NB:PRINT"CONVERSION PROCESS COMPLETE.  SUPPL
IES STAND AT";:IFSU-YR*2<.1PRINT"0 %"ELSEPRINTSU-YR*2;"%"
4900 REM
4990 GOTO 8000
5000 CLS:PRINTSHIPNA$;" STATUS REPORT":PRINT
5010 PRINT"POWER-";PWR;"%":PRINT"SUPPLIES-";SUPPLY;"%":PRINT"COL
ONISTS-";CLNSTS;"IN STASIS":PRINT"ENGINES- ";:IF EN$="Y" GOSUB 9
020 ELSE GOSUB 9030
5030 PRINT"SCANNERS- ";:IFSC$="Y"GOSUB9020ELSEGOSUB9030
5040 PRINT:GOSUB3010
5050 PRINT:PRINT"CURRENT DATE:";CD:PRINT"END OF MISSION:";2310+Y
E:GOTO 8000
7999 REM ACCT FOR TIME SPENT
8000 CDATE=CDATE+YR:PER=YR:YR=0:IFCDATE<2310+YEARSTHENGOTO8010EL
SEPRINT:PRINT"YOUR MISSION IS OVER, CAPTAIN ";NA$:CS=0:E=0:CF=0
8001 FORN=1TO32:IFST$(N,2)="E"THENE=E+1ELSEIFST$(N,2)="C"ANDPL(N
,3)>0THENCS=CS+1ELSEIFST$(N,2)="C"ANDPL(N,3)<1THENCF=CF+1
8002 NEXTN:PRINT:PRINT"YOU ENLARGED SOL'S SPHERE OF INFLUENCE TO
";CF+CS+E;"SYSTEMS":PRINT"SUCCESSFULLY COLONIZED";CS:PRINT"UNSUC
CESSFULLY COLONIZED";CF:PRINT"EXPLORED";E;"MORE.":PRINT
8003 IFCF+CS+E>17PRINT"OUTSTANDING JOB, ";NA$:PRINT"YOU ARE PROM
OTED IMMEDIATELY TO RANK OF ADMIRAL!";END
8004 IFCF+CS+E>10PRINT"NOT BAD, ";NA$;", BUT YOU COULD HAVE TRIE
D HARDER.":PRINT"RETAIN RANK WITH HONORABLE DISCHARGE."ELSEPRINT
"YOU ARE AN EMBARASSEMENT TO THE FLEET, ";NA$:PRINT"DEMOTED TO R
ANK OF ENSIGN WITH DISHONORABLE DISCHARGE!"
8008 END
8010 PWR=PWR-PER:IF PWR>5 GOTO 8020 ELSE PRINT"WE'VE RUN OUT OF
FUEL FOR OUR ENGINES AND GENERATORS.  WE'RE    DOOMED TO DRIFT F
OREVER!":END
8020 SU=SU-PE*2:IF SU>5 THEN GOTO 8030 ELSE PRINT"WE'VE RUN OUT
OF SUPPLIES, CAPTAIN.  THEN CREW HAS MUTINIED!    THEY'RE ON THE
IR WAY TO BOOT YOU OUT THE AIRLOCK...":END
8030 IF EN$="Y" GOTO 8040 ELSE IF CDATE>=EDATE THEN PRINT"THE EN
```

Program continues

Program continued

```
GINEERS HAVE FIXED THE JUMP ENGINES, SIR!  FULL JUMP      ABILIT
Y HAS BEEN RESTORED!":EN$="Y"
8040 REM
8900 C9$=CHR$(28):GOTO 1120
9000 PRINT@960,CHR$(30);:PRINT@960,"CAN'T GO OFF THE SCREEN.";:R
ETURN
9009 REM SPACES TO CORNER
9010 PRINT@768,SP$;:PRINT@832,SP$;:PRINT@896,SP$;:RETURN
9019 REM OP MSGS
9020 PRINT"FULLY OPERATIONAL":RETURN
9030 PRINT"REDUCED CAPACITY":RETURN
9049 REM GET RESPONSE Y OR N
9050 IF PEEK(14344)=2 THEN R=1 ELSE IF PEEK(14338)=64 THEN R=0 E
LSE GOTO 9050
9055 IF PEEK(14344) >< 2 AND PEEK(14338) >< 64 THEN RETURN ELSE
GOTO 9055
9199 REM JIGGLE SCREEN
9200 FOR N=1 TO 15:OUT255,8:FOR D=1 TO 3:NEXT D:OUT255,0:FOR D=1
 TO 3:NEXT D:NEXT N:RETURN
9224 REM FELL INTO A BLACK HOLE
9225 CLS:FOR N=1 TO 10:PRINT@RND(959),".";:NEXT N:PRINT@480,"*";
:M=400:FOR N=1 TO 20:PRINT@0,CHR$(23);:FOR D=1 TO 3:NEXT D:PRINT
@0,CHR$(28);:FOR D=1 TO 3:NEXT D:IF M>0 THEN M=M-40:FOR P=1 TO M
:NEXT P
9226 NEXT N:RETURN
9249 REM FIND DIST
9250 IF SQR(DX[2 + DY[2 + DZ[2) > MX THEN R=1 ELSE R=0
9251 RETURN
9269 REM FIND SYSTEM NAME
9270 FOR N=1 TO 32:IF ADDR=STAR(N,4) AND Z=STAR(N,3) THEN PRINT"
 - THE ";STAR$(N,1);" SYSTEM." ELSE NEXT N:PRINT
9271 RETURN
9299 REM JUMP SCREEN
9300 CLS::FOR N=1 TO 10:X1=RND(62):Y1=RND(13):PRINT@Y1*64+X1,"."
;:NEXT:FOR N=1 TO 200:NEXT:FOR N=1 TO 5:PRINT@0,CHR$(23);:FOR D=
1 TO 3:NEXT:PRINT@0,CHR$(28);:FOR D=1 TO 3:NEXT:NEXT N:CLS
9301 VA=176:GOSUB 9305:VA=32:GOSUB 9305:CLS:FOR N=1 TO 200:NEXT
N:RETURN
9305 X1=15838:X2=15840:X3=X1:X4=X2:FOR N=1 TO 6:X3=X3+60:X1=X1-6
8:X4=X4+68:X2=X2-60:POKE X1,VA:POKE X2,VA:POKE X3,VA:POKE X4,VA:
NEXT N:RETURN
9399 REM SYSTEM VALUE
9400 TYPE=STAR(SHIPLG(N,2),5)
9405 IF TYPE=98 THEN M3$="LOW GRADE METALS":M4$="NO FAUNAL/FLORA
L TYPES":RETURN ELSE IF TYPE=99 THEN M3$="NO VALUE WHATEVER":M4$
="": RETURN ELSE IF TYPE>10 THEN TYPE=TYPE-10
9410 IF TYPE=1 OR TYPE=2 OR TYPE=3 THEN M1$="LOW" ELSE IF TYPE=4
 OR TYPE=5 OR TYPE=6 THEN M1$="MED" ELSE IF TYPE=7 OR TYPE=8 OR
TYPE=9 THEN M1$="HIGH"
9420 IF TYPE=1 OR TYPE=4 OR TYPE=7 THEN M2$="LOW" ELSE IF TYPE=2
 OR TYPE=5 OR TYPE=8 THEN M2$="MED" ELSE IF TYPE=3 OR TYPE=6 OR
TYPE=9 THEN M2$="HIGH"
9430 M3$=M1$ + " GRADE METALS":M4$=M2$ + " LEVEL FAUNAL/FLORAL T
YPES"
9450 RETURN
9500 IFDD><INT(DD)PRINT:PRINT"MUST BE INTEGER":PRINT:R=0ELSER=1
9501 RETURN
```

ing from primitive to highly evolved. You may colonize any system that has been explored except hostile, inhabited systems. The only option in such a case is to leave quickly or to eventually be shot down.

If you choose to colonize a particular system, take note of the system's resources. If the system is poor, initially more supplies will be needed to start the colony. If you choose to force the colonists to land against their will, you may find them unreceptive to a return visit.

After the decision of colonization has been settled, you have the option of mining and refining the system's resources for future supplies.

The command Jump allows your ship to travel instantaneously from one point in space to another without crossing the intervening distance. This is the same method of travel that was used in "Star Guard." (Seven and a half lightyears is the maximum safe distance for jumping.) Jumping beyond this there is the danger of becoming lost by jumping to a random point in space along with blowing your ship's engines. In these situations, you will have to limp along as best you can until your engineers can repair the damage.

Status indicates in percentiles your reserve of supplies and fuel remaining, colonists left in stasis, and years left in the mission.

The Mission Begins

You feel a low rumble through the soles of your feet. Your shuttle has docked. As you wait for the air to cycle into the docking bay, you almost feel the weight of the entire world settling onto your shoulders. So much depends on your success! Grimly, you vow not to waste supplies or colonists. You are determined to explore and colonize as many of the best star systems as you can. The mission has begun. ■

✓See List of Advertisers on page 386

Interest-generating programs for the Pocket Computer.

Loan Sharp

Walter J. Atkins, Jr. Ph.D.
Qtrs 4410A
USAF Academy, CO 80840

Have you ever wondered what the monthly payment on a new car would be, or doubted that a salesman was quoting you the true interest rate? The Pocket Computer can help you.

I selected three programs ideally suited for the TRS-80 Pocket Computer from my collection of financial programs for my Model I. The first program finds the term of a loan. The second finds the loan payment amount. The third finds the true annual interest rate. I have combined these into one Pocket Computer program. Each of the program modules can also be used alone.

The Pocket Computer has a defined program mode that allows any of several programs in memory to be run by pressing two keys. If the computer is put into the defined program (DEF) mode, pressing Shift A runs the module that finds the term of a loan. Pressing Shift B runs the payment amount module, and Shift C runs the interest rate module.

The Program

The program takes about 800 steps and uses half the Pocket Computer's memory. The program is 40 lines long; it should be obvious to those of you who do not yet have a Pocket Computer that it can easily handle some very serious computing tasks.

The loan program is divided into three parts. Lines 10–70 find the time required to pay off a loan of a certain amount when the interest rate and amount of monthly payments are known. There is a minimum payment that must be made to pay off a loan. If a monthly payment of less than that minimum is made, the interest will exceed the payment and the loan will never be paid off. If a monthly payment less than the

- I—Annual interest rate in percent
- J—Last interest rate guess
- K—Interest rate increment
- N—Loan term in months
- P—Amount of loan
- Q—Monthly payment
- R—Computed payment in interest rate module
- Y—Loan term in years
- Z—Minimum monthly payment

Table 1. Variables

```
10 :REM *LOANTERM-SHFTA
15 :"A"
20 :PAUSE"FIND LOAN TERM"
25 :INPUT"LOAN AMOUNT=>?";P
30 :INPUT"ANN.INT.RATE(%)=>?";I
35 :Z=P*I/1200:Z=INT(Z*100+.5)/100
40 :PAUSE"TO PAY LOAN AT  ";I:PAUSE"PERCENT"
45 :PAUSE"MINIMUM PAYMENT=$";Z
50 :INPUT"MONTHLY PAYMENT=>?";Q
55 :Y=-(LOG(1-(P*(I/100))/(12*Q)))/(LOG(1+I/100/12)*12))
60 :N=Y*12:N=INT(N+.5)
65 :PRINT"LOAN TERM(MONTHS)=";N
70 :END
100: REM *LOANPAYMENT*
110:"B"
115:PAUSE"FIND PAYMENT AMOUNT"
120:INPUT"AMOUNT OF LOAN=>?";P
130:INPUT"ANN.INT.RATE(%)=>?";I:I=I/100
140:INPUT"NUMBER OF MONTHS=>?";N
150:Q=(I*P/12)/(1-((I/12+1)^-N))
160:Q=INT(Q*100+.5)/100
170:PRINT"MONTHLY PAYMENT=$";Q
180:END
200:REM *FINDRATE*
210:"C"
215:PAUSE"FIND INTEREST RATE"
220:INPUT"MONTHLY PAYMENT=>?";Q
230:INPUT"AMOUNT OF LOAN=>?";P
240:INPUT"NUMBER OF MONTHS=>";N
250:I=10:J=0
260:I=I/100:PAUSE"COMPUTING..PLS..WAIT"
270:R=(I*P/12)/(1-((I/12+1)^-N)):R=INT(R*100+.5)/100
280:K=ABS(I*100-J)/2:J=I*100
290:IF R=Q THEN 330
300:IF R>Q THEN 320
310:I=I*100+K:GOTO 260
320:I=I*100-K:GOTO 260
330:I=I*100:I=INT(I*1000+.5)/1000
340:PRINT"ANN.INT.RATE(%)=";I
350:END
```

Program Listing.

minimum is entered in the program, the Pocket Computer indicates an error when it tries to evaluate the LOG function in line 55. I have included lines 35–45 so the computer calculates and displays the minimum payment before it asks you to enter your monthly payment.

Lines 100–180 calculate the monthly payment required to pay off a loan of a certain amount in a specified period at a given interest rate.

Lines 200–350 calculate the annual interest rate. This section uses an iterative technique to determine the rate. First it assumes an interest rate of 10 percent. It then calculates the payment necessary to pay off the loan at that interest rate. If the calculated payment is higher than the monthly payment you entered, it assumes a lower interest rate and repeats the process. If the calculated payment is lower than your monthly payment, it assumes a higher interest rate and repeats the process. After a few iterations, the computer zeroes in on the true interest rate. The iteration process can take several seconds, so in line 260 I have the computer print out the message "COMPUTING..PLS..WAIT" after each iteration. ■

```
SHFT A
Find Loan Term
Loan Amount = >? 10000
ANN. INT. Rate(%) = >? 15
To Pay Loan At 15 Percent
Minimum Payment = $125
Monthly Payment = >?1234.56
Loan Term (Months) = 9

SHFT B
Find Payment Amount
Amount Of Loan = >? 1234.56
ANN. INT. Rate(%) = >? 10
Number Of Months = >? 24
Monthly Payment = $56.97

SHFT C
Find Interest Rate
Monthly Payment = >? 900
Amount Of Loan = >? 85000
Number Of Months = >? 360
ANN. INT. Rate(%) = 12.391
```

Table 2. Sample Runs

✓See List of Advertisers on page 386

Tired of abbreviations for error messages? Read on.

Full Error

Harry and Ken Keairns
920 La Plante
Sioux City IA 51109

While friends with expansion interfaces and disk drives enjoyed full error messages, we struggled with the two letter abbreviations characteristic of Level II Basic. After almost wearing out the Basic Reference Manual, we set out to find a way to develop longer messages on our system, without spending the money for a disk system.

Our first effort produced the Basic program in Program Listing 1. We generally loaded this before we started developing a program, and after the program was debugged, replaced it with an error handling routine. While it served our initial purpose, it was inconvenient to load it for every program.

It gave us a better understanding of the ERL and ERR/2 + 1 functions, which are important when developing an error routine. The ERL function returns the line number that contains an error, and the ERR/2 + 1 function returns the error number that corresponds with the table on page B/1 of the manual.

RAM Link

Like many TRS-80 users, we spend time wandering around the ROM area. On one of these excursions we discovered that memory location 41A6H provides a link between the Level II error routine and the RAM area of memory. This location is called both before the error message is sent to the screen and immediately afterward.

Program Listing 2 disassembles this area of ROM (19FEH thru 1A1EH), with remarks on the portion we will be changing. The link location (41A6H) normally contains a Return statement, but provision was made for a three-byte Jump command to another location in memory. We will use this provision to replace the ROM routine with one of our own design.

Replace the first 16 bytes of the ROM routine, and use the balance of the routine to send the remainder of the message (i.e., "Error in line nn").

The first section of Program Listing 3 is a short routine that replaces the Return statement at 41A6H with a jump to the start of our routine. After loading the object code tape, entering a start point of 32292 initializes the new subroutine. This will have to be repeated in the event of a system crash, but can be protected by answering the memory size question with 32291 when you first bring the '80 to life.

The second section defines the messages to be sent to the screen in place of the two-letter abbreviations. Note that the messages end with a CHR$(34), as the ROM routine used to get the message to the screen looks for this as a delimiter. When it finds a quotation mark (or a zero byte), it knows the message is complete.

The next section is a table of the addresses for the messages previously defined. This approach is similar to the approach used by the ROM routine, and it solves the problem of messages of unequal length.

The final section selects the message, sends it to the screen, and then completes the message using the original ROM routine.

The actual selection is handled by loading the HL registers with the first address in our two-byte address table. The ERR code is then added, and the HL registers are pointed at the proper message. Make use of the DE registers to retrieve the address—save their contents with a PUSH instruction. After loading the address bytes, exchange DE and HL, as the routine expects to find the address there. Restore DE with a POP and the message is ready to be sent to the screen.

Now make a CALL to 28A7H, the location of a ROM routine that displays a string on the

✓See List of Advertisers on page 386

screen. After sending the string revert to the ROM routine to send the word "error," and the line number, if appropriate. End the routine with a Jump to 1A1FH, the location immediately after the second call to 41A6H.

This program uses less than 500 bytes and usually saves much more than the eight-second loading time in look-up time alone. We also hope it will provide the ROM-roamers amongst us with new locations to experiment with.■

```
0  ON ERROR GOTO 24 : GOTO 25
1  PRINT@64, "NEXT WITHOUT FOR ERROR IN LINE" ; ERL : END
2  PRINT@64, "SYNTAX ERROR IN LINE" ; ERL : END
3  PRINT@64, "RETURN WITHOUT GOSUB ERROR IN LINE" ; ERL : END
4  PRINT@64, "OUT OF DATA ERROR IN LINE" ; ERL : END
5  PRINT@64, "ILLEGAL FUNCTION CALL IN LINE" ; ERL : END
6  PRINT@64, "OVERFLOW ERROR IN LINE" ; ERL : END
7  PRINT@64, "OUT OF MEMORY ERROR IN LINE" ; ERL : END
8  PRINT@64, "UNDEFINED LINE IN LINE" ; ERL : END
9  PRINT@64, "SUBSCRIPT OUT OF RANGE ERROR IN LINE" ; ERL : END
10 PRINT@64, "REDIMENSIONED ARRAY ERROR IN LINE" ; ERL : END
11 PRINT@64, "DIVISION BY ZERO ERROR IN LINE" ; ERL : END
12 REM       'ID ERROR POSSIBLE COMMAND MODE ONLY
13 PRINT@64, "TYPE MISMATCH ERROR IN LINE" ; ERL : END
14 PRINT@64, "OUT OF STRING SPACE ERROR IN LINE" ; ERL : END
15 PRINT@64, "STRING TOO LONG ERROR IN LINE" ; ERL : END
16 PRINT@64, "STRING FORMULA TOO COMPLEX IN LINE" ; ERL : END
17 PRINT@64, "CAN'T CONTINUE ERROR IN LINE" ; ERL : END
18 PRINT@64, "NO RESUME ERROR IN LINE" ; ERL : END
19 PRINT@64, "RESUME WITHOUT ERROR IN LINE" ; ERL : END
20 PRINT@64, "UNPRINTABLE ERROR IN LINE" ; ERL : END
21 PRINT@64, "MISSING OPERAND ERROR IN LINE" ; ERL : END
22 PRINT@64, "BAD FILE DATA ERROR " : END
23 PRINT@64, "DISC BASIC ERROR " : END
24 ON ERR/2+1 GOTO 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
 18,19,20,21,22,23
25 REM       'START YOUR PROGRAM HERE
```

Program Listing 1.

```
19EC                ORG     19ECH
19EC CDA641         CALL    41A6H          ;RAM LINK LOCATION
19EF 57             LD      D,A            ;PUT ERROR CODE IN D
19F0 3E3F           LD      A,3FH          ;QUESTION MARK
19F2 CD2A03         CALL    032AH          ;DISPLAYS SINGLE BYTE
19F5 19             ADD     HL,DE          ;POINTS TO TYPE OF ERROR
19F6 7E             LD      A,(HL)         :GET FIRST LETTER
19F7 CD2A03         CALL    032AH          :DISPLAY LETTER
19FA D7             RST     10H            :INC HL AND  LOADS 2ND LETTER
19FB CD2A03         CALL    032AH          :DISPLAY 2ND LETTER
19FE 211D19         LD      HL,191DH       ;POINTS TO WORD 'ERROR'
1A01 E5             PUSH    HL
1A02 2AEA40         LD      HL,(40EAH)
1A05 E3             EX      (SP),HL
1A06 CDA728         CALL    28A7H
1A09 E1             POP     HL
1A0A 11FEFF         LD      DE,0FFFEH
1A0D DF             RST     18H
1A0E CA7406         JP      Z,0674H
1A11 7C             LD      A,H
1A12 A5             AND     L
1A13 3C             INC     A
1A14 C4A70F         CALL    NZ,0FA7H
1A17 3EC1           LD      A,0C1H
1A19 CDBB03         CALL    03BBH
1A1C CDAC41         CALL    41ACH
0000                END
00000 TOTAL ERRORS
```

Program Listing 2.

```
             ORG     7E24H
 ;*********************************************************
 ;*                                                       *
 ;*   THIS SECTION INITIALIZES RAM LINK                   *
 ;*                                                       *
 ;*********************************************************
 ;
             LD      HL,41A6H        ;POINT TO RAM LINK
             LD      DE,START        :FIND START
             LD      (HL),0C3H       :JUMP TO -
             INC     HL
             LD      (HL),E          :- LSB
             INC     HL
             LD      (HL),D          : - MSB
             JP      0072H           ;BACK TO BASIC
 ;*********************************************************
 ;*                                                       *
 ;*   THIS SECTION DEFINES THE ERROR MESSAGES             *
 ;*                                                       *
 ;*********************************************************
 ;
 NFERR       DEFM    'NEXT WITHOUT FOR"'
 SNERR       DEFM    'SYNTAX"'
 RGERR       DEFM    'RETURN WITHOUT GOSUB"'
 ODERR       DEFM    'OUT OF DATA"'
 FCERR       DEFM    'ILLEGAL FUNCTION CALL"'
 OVERR       DEFM    'OVERFLOW"'
 OMERR       DEFM    'OUT OF MEMORY"'
 ULERR       DEFM    'UNDEFINED LINE"'
 BSERR       DEFM    'SUBSCRIPT OUT OF RANGE"'
 DDERR       DEFM    'REDEMENSIONED ARRAY"'
 DOERR       DEFM    'DIVIDE BY ZERO"'
 IDERR       DEFM    'ILLEGAL DIRECT COMMAND"'
 TMERR       DEFM    'TYPE MISMATCH"'
 OSERR       DEFM    'OUT OF STRING MEMORY"'
 LSERR       DEFM    'STRING TOO LONG"'
 STERR       DEFM    'STRING TOO COMPLEX"'
 CNERR       DEFM    'CAN NOT CONTINUE"'
 NRERR       DEFM    'NO RESUME"'
 RWERR       DEFM    'RESUME WITHOUT"'
 UEERR       DEFM    'UNPRINTABLE"'
 MOERR       DEFM    'MISSING OPERAND"'
 FDERR       DEFM    'BAD FILE DATA"'
 L3ERR       DEFM    'DISC BASIC ONLY"'
 ;
 ;*********************************************************
 ;*                                                       *
 ;*  THIS SECTION IS A TABLE OF MESSAGE ADDRESSES         *
 ;*                                                       *
 ;*********************************************************
 ;
 TABL        DEFW    NFERR
             DEFW    SNERR
             DEFW    RGERR
             DEFW    ODERR
             DEFW    FCERR
             DEFW    OVERR
             DEFW    OMERR
             DEFW    ULERR
             DEFW    BSERR
             DEFW    DDERR
             DEFW    DOERR
             DEFW    IDERR
             DEFW    TMERR
             DEFW    OSERR
             DEFW    LSERR
             DEFW    STERR
             DEFW    CNERR
             DEFW    NRERR
             DEFW    RWERR
             DEFW    UEERR
             DEFW    MOERR
             DEFW    FDERR
             DEFW    L3ERR
 ;
 ;*********************************************************
 ;*                                                       *
 ;*   THIS SECTION SELECTS THE CORRECT MESSAGE            *
 ;*                                                       *
 ;*********************************************************
 ;
 START       LD      HL,TABL         ;POINT TO LOOK-UP TABLE
             ADD     HL,BC           ;ADVANCE TO MESSAGE NEEDED
             PUSH    DE              ;SAVE REGISTER
             LD      E,(HL)          ;GET LSB OF MESSAGE ADDRESS
             INC     HL
             LD      D,(HL)          ;GET MSB OF MESSAGE ADDRESS
             EX      DE,HL           ;POINT TO  MESSAGE
             POP     DE              ;RESTORE REGISTER
 ;
 ;*********************************************************
 ;*                                                       *
 ;*   THIS SECTION SENDS MESSAGE TO THE SCREEN            *
 ;*                                                       *
 ;*********************************************************
 OUTPT       CALL    28A7H           ;DISPLAY STRING TO SCREEN
             LD      HL,191DH        ;POINT TO 'ERROR' MESSAGE
             PUSH    HL              ;SAVE IT
             LD      HL,(40EAH)
             EX      (SP),HL
             CALL    28A7H
             POP     HL
             LD      DE,0FFFEH
             RST     18H
             JP      Z,0674H
             LD      A,H
             AND     L
             INC     A
             CALL    NZ,0FA7H
             LD      A,0C1H
             CALL    03BBH
             JP      1A1FH
             END
```

Program Listing 3.

✓See List of Advertisers on page 386

Store source and object code on disk with this Microsoft program.

Modifying EDTASM Plus

Arne Rohde
Pilevej 31
7600 Struer
Denmark

One of the main problems with converting from a tape-based TRS-80 to disk drives is that many programs use tape for data storage. Conversion is usually no problem with Basic programs, since the user has access to the source code and relocation is handled by the Basic interpreter. System programs are often more problematic. Relocation is complicated, if the source code is not available, and is desirable, if DOS facilities are needed and the program storage area conflicts with the DOS storage area.

The common solution is to move the program in memory so it can be loaded with DOS, then move it to its correct location, destroying the DOS. The DOS facilities, such as data storage on disk, are no longer available, and the user is forced to continue to use tape for this purpose.

Microsoft Editor/Assembler Plus

Microsoft's Editor/Assembler Plus is tape-based. It was intended to replace Radio Shack's EDTASM and T-Bug, offering more facilities at a lower price. The disadvantage is that no modification is available allowing it to be used on a disk-based system to store source and object code on disk.

Apparat Inc. has made a modification to Radio Shack's EDTASM which allows it to be used with TRSDOS or NEWDOS. It comes with NEWDOS/80, but the user must purchase the Radio Shack version to obtain the documentation. There are still advantages to using the Microsoft version, such as a macro facility, the ability to generate code directly into memory, and the option of leaving the source in memory while the program is debugged with Z-Bug.

I have used both and prefer the Microsoft version for most purposes, even though it lacks a cross-reference listing facility. The program itself uses almost 12K storage when all modules are resident, so the sheer size of the task puts relocation out of the question for me. However, relocation is the only way to store source code on disk. If the Dump command could be used to move the source to disk and Load used to reload it into memory before the assembler is moved into position, the source code need not be stored on tape. Since object code can be generated directly into memory (as long as it does not conflict with the assembler), Dump can also be used to store the object code on disk. In certain cases it may be necessary to write the object code to tape, and then copy it to disk with TAPEDISK or similar utility.

Modifying EDTASM/Plus

I have not been able to find any free memory within EDTASM/Plus, so the modifications must be made by removing commands and using up the resulting space by appending the utility onto the end of the code and forcing the program to regard this as reserved memory or by locating the utility at the high end of memory and forcing the program to avoid this area.

I rejected the first approach, since none of the commands seem dispensable. The third approach requires different procedures for different memory sizes and restricts the routines which can be located in this area. I chose the second approach, even though it has one major disadvantage—quashing the assembler or Z-Bug destroys the modification. However, if quashing is necessary, the advantages of having all three routines available simultaneously would no longer apply. Since I have 48K available (over 32K free), large programs can usually be written without a need to quash.

EDTASM/Plus occupies the area from 4380H - 7265H, but source code is written from 71C9H, the remainder being one-time code. I use the area from 7266H - 72FFH, allowing the source code to begin at 7300H. I also use the area normally used by the Quash command. When I store programs such as EDTASM/Plus on disk, I save more memory than is necessary in order to recreate the original program environment as closely as possible without

leftovers from DOS. I do this by storing the contents of reserved memory from 4000H to the beginning of the program, together with the program itself. The modifications shown in the Program Listing can be applied by using the following sequence, using RSM2D as the monitor.

1. Load RSM2D.
2. Reset with the break key held down.
3. Answer the memory size question with a specific value.
4. Type System.
5. Load EDTASM/Plus.
6. Enter RSM2D by typing/entry address.
7. Type in the changes shown in the Listing. (You must supply the hex values of the text strings.)
8. Move the area 4000H-7305H to 5200H-8505H.
9. Press reset to load the DOS.
10. Dump the area from 5200H - 8505H to disk, with either 84F0H or no transfer point.
11. If DOS transfer is used, Dump 8500H - 8505H, and transfer 84F0H as a dummy source module.

The procedure for initiating the assembler (assuming DOS entry point) is now as follows:

1. Type EDTASM (if EDTASM /Plus is stored as EDTASM/ CMD).
2. Type the name of the source file to be loaded.

If 84F0H is used as transfer point for EDTASM, Load must be used for EDTASM when a source file is to be loaded immediately afterward.

When a source file is to be saved on disk, command Q is used (for Quit, instead of Quash) to move the source in memory and to display the values to be used for the Dump statement. Note these values since the screen will be erased as soon as the space key is depressed. The Dump command should be used as soon as DOS has been loaded. The source files are *not* directly transferable to other assemblers, although a conversion program could easily be written. The source could also be transferred via cassette tape.

Comments

The modifications may need a few comments. I found that the Q command resides in the area 646CH - 64B3H. The command table address starts at 4659H, with one byte for the command code and two bytes for the address of the corresponding routine. These values should be checked before the modifications are applied. I used the EDTASM/Plus warm entry point (4383H), since source code is present on entry. This entry skips the check for the end-of-memory address (normally all memory is used with no check for reserved memory), so this address must also be set up. I use this opportunity to reserve memory to advantage. I have a driver for an RS232 printer resident here, so it may be used with EDTASM/Plus. Also, I leave room for RSM2D or other code as required.

The source start and end addresses are stored together with the source code in the four bytes at 8500H. When command Q is used, the source code and addresses are moved in memory, but no check is made for memory overflow. A check could be made that the new end address does not exceed the address in location 4236H, though it should

Program Listing.

```
                  00100 ;CHANGES TO MICROSOFT EDTASM-PLUS TO ALLOW FOR STORING
                  00110 ;ASSEMBLER TEXT ON DISK
                  00120 ;PROGRAMMED BY ARNE ROHDE, STRUER, DENMARK, SEPTEMBER 1980
                  00130 ;COMMAND Q USED TO EXIT AND MOVE SOURCE IN MEMORY
8500              00140 BFSTAD  EQU     8500H           ;STORE BUFFER START ADDR
8502              00150 BFENAD  EQU     BFSTAD+2        ;STORE BUFFER END ADDR
8504              00160 BUFFER  EQU     BFENAD+2        ;STORE TEXT BUFFER
EBFF              00170 MEMEND  EQU     0EBFFH          ;MEMORY END, ROOM FOR RSM2D
                  00180                                 ;PRINTER DRIVER, ETC
646C              00190         ORG     646CH           ;Q COMMAND AREA
646C 2A3242       00200         LD      HL,(4232H)      ;END OF SOURCE PTR
646F E5           00210         PUSH    HL              ;STORE END ADDR
6470 ED5B3042     00220         LD      DE,(4230H)      ;START OF SOURCE
6474 D5           00230         PUSH    DE              ;STORE START ADDR
6475 23           00240         INC     HL              ;INCLUDE FFFF END MARK
6476 E5           00250         PUSH    HL              ;STORE ADDR
6477 23           00260         INC     HL
6478 AF           00270         XOR     A               ;CLR CARRY
6479 ED52         00280         SBC     HL,DE           ;FIND MOVE LENGTH
647B 44           00290         LD      B,H
647C 4D           00300         LD      C,L             ;XFER TO BYTE COUNT
647D 210385       00310         LD      HL,BUFFER-1     ;DEST ADDR -1
6480 09           00320         ADD     HL,BC           ;+ LEN = END ADDR
6481 EB           00330         EX      DE,HL           ;DEST ADDR TO DE
6482 E1           00340         POP     HL              ;SOURCE ADDR
6483 D5           00350         PUSH    DE              ;STORE END ADDR
6484 EDB8         00360         LDDR                    ;MOVE TEXT
6486 D1           00370         POP     DE              ;END ADDR AGAIN
6487 E1           00380         POP     HL              ;START ADDR
6488 220085       00390         LD      (BFSTAD),HL     ;STORE WITH TEXT
648B E1           00400         POP     HL              ;END ADDR
648C 220285       00410         LD      (BFENAD),HL     ;STORE IT AS WELL
648F 219C72       00420         LD      HL,TEXT+37      ;END ADDR TEXT
6492 7A           00430         LD      A,D
6493 CD6672       00440         CALL    CNVL            ;CONVERT MSD TO HEX
6496 7A           00450         LD      A,D
6497 CD6A72       00460         CALL    CNVR            ;CONVERT LSD TO HEX
649A 7B           00470         LD      A,E
649B CD6672       00480         CALL    CNVL            ;CONVERT MSD TO HEX
649E 7B           00490         LD      A,E
649F CD6A72       00500         CALL    CNVR            ;CONVERT LSD TO HEX
64A2 217772       00510         LD      HL,TEXT
64A5 CD3245       00520         CALL    4532H           ;WRITE TO SCREEN
64A8              00530 WTENT   EQU     $
64A8 3A4038       00540         LD      A,(3840H)       ;KEYBOARD MEMORY
64AB E680         00550         AND     80H             ;ISOLATE SPACE
64AD 28F9         00560         JR      Z,WTENT         ;NOT SPACE KEY
64AF C30000       00570         JP      0               ;BOOT SYSTEM
                  00580 ;COULD CONT TO 64B3H, END OF Q COMMAND
                  00590 ;
                  00600 ;END OF EDTASM - EXTRA MEMORY RESERVED
7266              00610         ORG     7266H
                  00620 ;CONVERT NIBBLE TO HEX CHARACTER
7266              00630 CNVL    EQU     $
7266 0F           00640         RRCA
7267 0F           00650         RRCA
7268 0F           00660         RRCA
7269 0F           00670         RRCA                    ;MOVE MSD TO LSD
726A              00680 CNVR    EQU     $
726A E60F         00690         AND     0FH             ;REMOVE UNWANTED
726C F630         00700         OR      30H             ;ASSUME NUMERIC
726E FE3A         00710         CP      3AH             ;CHECK ALPHA
7270 3802         00720         JR      C,HEXOK
7272 C607         00730         ADD     A,07H           ;CONVERT TO A-F
7274              00740 HEXOK   EQU     $
7274 77           00750         LD      (HL),A          ;STORE IN TEXT
7275 23           00760         INC     HL
7276 C9           00770         RET
7277 0A0A         00780 TEXT    DEFW    0A0AH           ;LF LF
7279 44           00790         DEFM    'DUMP FILENAME (START=X'
```

Program Listing continues

Program Listing continued

```
728F 27       00800          DEFB    27H
7290 38       00810          DEFM    '8500'
7294 27       00820          DEFB    27H
7295 2C       00830          DEFM    ',END=X'
729B 27       00840          DEFB    27H
729C 58       00850          DEFM    'XXXX'
72A0 27       00860          DEFB    27H
72A1 2C       00870          DEFM    ',TRA=X'
72A7 27       00880          DEFB    27H
72A8 38       00890          DEFM    '84F0'
72AC 27       00900          DEFB    27H
72AD 29       00910          DEFM    ')'
72AE 0A       00920          DEFB    0AH                ;LINE FEED
72AF 28       00930          DEFM    '(SPACE) TO BOOT'
72BE 0A       00940          DEFB    0AH                ;LF
72BF 80       00950          DEFB    80H                ;TERMINATE TEXT
              00960 ;
              00970 ;ENTRY POINT AFTER EDTASM MOVED IN MEMORY
72C0          00980 BEGIN    EQU     $
72C0 318043   00990          LD      SP,4380H           ;STACK FOR EDTASM
72C3 2A0085   01000          LD      HL,(BFSTAD)        ;TEXT START ADDRESS
72C6 223042   01010          LD      (4230H),HL         ;STORE IN EDTASM
72C9 E5       01020          PUSH    HL
72CA 2A0285   01030          LD      HL,(BFENAD)        ;TEXT END ADDR
72CD 223242   01040          LD      (4232H),HL         ;STORE IT
72D0 23       01050          INC     HL
72D1 23       01060          INC     HL                 ;PAST FFFF TERMINATOR
72D2 D1       01070          POP     DE                 ;DEST FOR MOVE
72D3 ED52     01080          SBC     HL,DE              ;LENGTH FOR MOVE
72D5 44       01090          LD      B,H
72D6 4D       01100          LD      C,L                ;STORE IN BYTE COUNT
72D7 210485   01110          LD      HL,BUFFER          ;START OF MOVE
72DA EDB0     01120          LDIR                       ;MOVE TEXT TO POSITION
72DC 21FFEB   01130          LD      HL,MEMEND          ;EDTASM END ADDR (MODIFY)
72DF 223642   01140          LD      (4236H),HL
72E2 2B       01150          DEC     HL
72E3 223442   01160          LD      (4234H),HL         ;STORE EDTASM POINTERS
72E6 C38343   01170          JP      4383H              ;ENTRY TO LEAVE TEXT BUFFER
              01180 ;
              01190 ;CODE ON ENTRY WHEN TEXT LOADED
72F0          01200          ORG     72F0H
72F0 F3       01210          DI                         ;DISABLE INTERRUPTS
72F1 010633   01220          LD      BC,3306H           ;MOVE LENGTH
72F4 210052   01230          LD      HL,5200H           ;BUFFER FROM
72F7 110040   01240          LD      DE,4000H           ;DEST ADDR
72FA EDB0     01250          LDIR
72FC C3C072   01260          JP      BEGIN              ;GO TO MOVE TEXT
72FF 00       01270          NOP                        ;FOR 7300H
              01280 ;SET UP ADDRESSES AND TEXT END MARKER FOR EMPTY TEXT BUFFER
              01290 ;REQUIRED SINCE WARM ENTRY USED, IF NO SOURCE MODULE LOADED
              01300 ;WILL BE OVERLAID IF SOURCE MODULE LOADED.
7300 0073     01310          DEFW    7300H              ;TEXT START
7302 0073     01320          DEFW    7300H              ;TEXT END
7304 FFFF     01330          DEFW    0FFFFH             ;END MARKER
0000          01340          END
```

be obvious from the values in the displayed message. The initial values stored at 7300H are used to initialize the source program area if no source program is to be loaded; they will be overlaid if a source program is loaded before executing the assembler.

The solution I chose is not ideal for converting programs to run on DOS systems, but it is reasonably easy to implement. The alternative would be to purchase a more expensive macro assembler and forget all about Z-Bug or to attempt both a relocation and a modification of EDTASM/Plus to allow it to be resident with DOS. With those alternatives, you can easily live with a few shortcomings!

I implemented the modifications using NEWDOS/80 and RSM2D, but I can see no reason why they should not work with TRSDOS as well. In fact, the format of the message with the Dump parameters has been coded in the TRSDOS format rather than the NEWDOS/80 format, which, incidentally, is much simpler.

The principles of these modifications are also applicable to other programs; I have converted Tiny Pascal to work with disk storage for source and P-code. It is now much faster and it is much more reliable than using tape.■

Not all Spelling Checkers are the same.

MICROPROOF[T.M.] stands out!

EASY TO USE: Prepare your text on any Z-80 based microcomputer, using any of a number of popular word processing programs. When you are finished, enter the appropriate command, and MICROPROOF proofreads your document, displaying misspellings and typos on the screen. Then correcting MICROPROOF can display each error separately, requesting you to enter the correct spelling for each. You are also given the option of displaying errors in context or adding words to MICROPROOF's 50,000 word vocabulary. Finally, MICROPROOF corrects your document. All in less than a minute.

```
SELECT APPROPRIATE RESPONSE:

CORRECT MISSPELLED WORD:    ENTER CORRECT WORD
LEAVE WORD "AS IS":         HIT <ENTER> KEY
DISPLAY WORD IN CONTEXT:    ?
ADD WORD TO DICTIONARY:     +
EXIT:                       !

WORD:        (Your error)
RESPONSE:
```

Correcting MICROPROOF Screen Display

SPEED is the single most important factor in a dictionary program. All dictionary programs will find your potential errors but if the program is too slow, you are not likely to use it. MICROPROOF's speed is outstanding. It can proofread a several page letter in 20 seconds.

LOW PRICES: Standard MICROPROOF is available for either \$89.50 (TRS-80® Models I or III) or \$149.50 (CP/M®, TRS-80® Model II and all others). The optional correction feature can be added at any time for an additional \$60.00. Optional patches to integrate MICROPROOF into your word processing software can also be added at any time for an additional \$35.00. (Integration patch not needed for Wordstar®.)

MICROPROOF'S FULL 50,000 WORD VOCABULARY saves you time and allows you greater confidence in the lists of potential errors that MICROPROOF identifies. The mini-dictionary programs, with their 10,000 and 20,000 word vocabularies, have many correctly spelled words omitted from their vocabularies. Consequentially, they identify as potential "errors" many words that are actually spelled correctly; five to ten times as many such words as does MICROPROOF. So, when you use MICROPROOF you will have far fewer extra words to evaluate, a major time savings. There will be less need to look up words in order to verify that they are in fact spelled correctly. The extra 30,000 words in MICROPROOF's vocabulary assures you confidence in the error lists that MICROPROOF generates.

There are other proofreading programs available to choose from. Since MICROPROOF became available in December of 1980, a number of companies have announced programs with small dictionaries. It took us almost two years to develop MICROPROOF. During that time we were able to compress our full 50,000 word dictionary into a manageable size (fits on one single density 5¼ inch disk). And we were able to design a proofing program which operates remarkably fast. The chart below illustrates the comparative advantages of MICROPROOF.

ADVANTAGES OF MICROPROOF

	MICROPROOF DICTIONARY SOFTWARE	OTHERS (Mini-Dictionaries)
DICTIONARY SIZE	50,000 Words	20,000 Words
DISK SPACE REQUIRED FOR DICTIONARY	70,000 BYTES (fits easily on one 5¼" disk)	170,000 BYTES
DICTIONARY ENLARGEMENT	VIRTUALLY UNLIMITED	EXTREMELY LIMITED
SPEED—400 Words	20 Seconds	1 to 5 Minutes
SPEED—3,000 Words	1 Minute	2 to 10 Minutes
CORRECTION FEATURE	Optional	Not Available

See your local microcomputer dealer or write to:

CORNUCOPIA SOFTWARE

Post Office Box 5028, Walnut Creek, California 94596 • (415) 893-0633 ✓378


GRAY MORROW

Get yourself a real chill on HALLOWEEN in the nationwide CompuServe Adventure Tournament.

Prizes: Grand Masters win two hours of free time on CompuServe.

Runners-up win a CompuServe Adventure T-shirt (Grand Masters win this too.)

Everyone who enters can, on request, receive a large poster of the Gray Morrow art on the opposite page in full color. We'll charge your account $2.00 for postage and handling.

When? Starting at 6 PM local time on Friday, October 30, running through 5 AM on Monday, November 2.

How? Since this will be our first nationally-advertised tournament, we may have to make some last minute changes following our "in-house" tournament on Labor Day. So check the "What's New" section of the CompuServe Information Service for details preceding the contest. So enter. Collect treasures. Slay dwarves. And practice, practice, practice. How else will we be able to afford the T-shirts?

CompuServe ✓424

5000 Arlington Centre Blvd.
Columbus, Oh 43220
Information Service Division
(800) 848-8990

If you're not already a member of the country's largest personal computer network, get on-line now!

How does it work?
Your computer talks to our computers. You need a modem and some special software. Software for the TRS-80® and Apple II® computers are available at many Radio Shack® outlets and all Radio Shack® Computer Centers. Atari® users can access the system via Telelink™ software. Persons owning other brands of computers or terminals can also purchase the CompuServe Information Service through Radio Shack®. In more than 260 U.S. cities you can reach the CompuServe Information Service via a local phone call. The cost is $5 per hour, billed in minutes, to your charge card, for service between 6 PM and 5 AM local time on weekdays and all day on Saturday, Sunday and most holidays. Limited daytime access is available at a higher rate.

What do I get?
There are more than 175 topics listed in the on-line index. Most first-time users are interested in:

1. News, weather, sports. We offer the electronic editions of many of the major daily newspapers and the AP news and sports wires.
2. CB simulation. "Talk" to other computer users anywhere in the country. We also offer electronic mail.
3. Games. Many, including the multi-player Space War, Star-Trek, four Adventure games.
4. Finance. Again, a wide variety for both historical and current data on stocks, bonds and commodities.
5. Computer user groups (including national bulletin boards) and computer manufacturers' newsletters.
6. Home information. Family Service. Government publications. Aviation news. Energy-saving tips. And more.

When you are ready, CompuServe Information Service can provide you with a lot of big mainframe computer power. But see a demonstration at Radio Shack® right away so you can get on-line before the tournament.

✓See List of Advertisers on page 386

An easier way for data entry.

Formatted Screen Input

F. Christian Byrnes
83 Fairmount Ave.
Hackensack, NJ 07601

Entering moderate volumes of data into the TRS-80 can be frustrating. A Basic program displays a prompting message which requests the input of one or more fields of data. If more than one field is to be input, the operator must know to use a comma between them.

Terminals attached to large systems enjoy the advantage of formatted screens. The operator sees the screen as being divided into headings or titles and input fields. Data corresponding to each heading is entered following its heading and the cursor jumps from one field to the next as the data is entered. All of the headings remain unaltered in place until all of the data for that screen have been entered.

Initially the TRS-80 does not appear to allow us the luxury of a formatted screen, but a little fancy footwork can change the operator's view of the computer drastically.

Overview

This program listing is an example of the formatting and input routines imbedded in a driver program to demonstrate how they may be used. Screen formatting is performed by the routine in statements 1100 and 1110 using arrays OS and OS$. Array OS contains, in element zero, the number of field headings in this screen. Each element of the array thereafter contains the screen starting position of a heading. The elements of array OS$, starting with element one, contain the text corresponding to the positions specified in OS.

Input is accepted by a call to the routine at statement 900. This routine selects the next element from array OM and calls the subroutine at statement 1000 to position the cursor and read the keyboard. Array OM has two dimensions. Each element pair consists of a starting position and length for an input field. Element (0,0) contains a count of input fields for this screen. The accepted data is returned in variable OB$. Variable OE contains a pointer to the next element of OM.

Logic

The example program shows how to use these subroutines. Line one dimensions the required arrays. Line 10 and 20 load these arrays from data statements. In a disk system this information could easily be read from a file.

Line 30 calls the screen formatting subroutine. Upon return all of the headings have been placed in their proper positions on the screen.

Line 35 calls the screen input procedure at line 900. Line 910 tests variable OE. The number of the next array element of OM to be used as location and length of the input field is contained in OE. The calling program could change this value prior to a call in order to skip or repeat a field. If such a function is to be used, a device such as T-Beep would be needed to alert the operator to the change.

Lines 920 and 930 set internal variables XP and XL equal to the referenced elements of array OM and are followed by a call to the subroutine at line 1000. This method was chosen so multiple format and input tables could be used within a single program. Modified versions of lines 900 to 960 could be coded in a single program, each for a different screen format, and all could call line 1000 for input.

- 1000 begins the keyboard input routine.
- 1001 sets OB$ to null.

```
1 CLEAR1500:DIM OM(25,2),OS(50),OS$(50)
5 DIM AA$(25)
10 READ OM(0,0):FOR OH=1 TO OM(0,0):READ OM(OH,1),OM(OH,2):NEXT
20 READ OS(0):FOR OH=1 TO OS(0):READ OS(OH),OS$(OH):NEXT
30 GOSUB 1100
35 GOSUB 900
40 AA$(OE-1)=OB$
50 IF OE<=OM(0,0) THEN 35
60 CLS:PRINT "THANK YOU"
70 'PROCESSING ROUTINE GOES HERE
80 FOR N=1 TO OE:PRINT AA$(N);"      ";:NEXTN
85 N$=INKEY$:IF LEN(N$)=0 THEN 85
90 GOTO 30
900 'INPUT ARRAY DRIVER
910 IF OE=0 OR OE>OM(0,0) THEN OE=1
920 XP=OM(OE,1)
930 XL=OM(OE,2)
940 GOSUB 1000
950 OE=OE+1
960 RETURN
1000 'GET FIELD
1001 OB$=""
1002 PRINT@XP,CHR$(14);
1004 FOR OC=1 TO XL
1005 OA$=INKEY$:IF LEN(OA$)=0 THEN 1005
1010 IF OA$=CHR$(8) THEN GOSUB 1050:GOTO1005
1012 IF OA$=CHR$(9) THEN OA$=CHR$(32)
1015 IF OA$=CHR$(13) THEN OC =XL:GOTO 1030
1019 IF OA$>CHR$(90) OR OA$<CHR$(32) THEN 1005
1020 OB$=OB$+OA$:PRINT OA$;
1030 NEXT:PRINT CHR$(15);:RETURN
1050 ' BACKSPACE KEY PRESSED
1055 IF LEN(OB$)=0 THEN OC=1:RETURN
1060 OB$=LEFT$(OB$,LEN(OB$)-1):PRINT CHR$(8);
1065 OC=OC-1
1070 RETURN
1100 CLS:FOR OH=1 TO OS(0):PRINT@OS(OH),OS$(OH):NEXT
1110 RETURN
9000 DATA 15,208,40,272,44,336,15,359,2,367,5,459,3,463,3,467,4,
476,4,525,40,671,8,735,8,841,55,905,55,969,55
9010 DATA 11,81,SAMPLE DATA COLLECTION SCREEN,192,ACCOUNT NAME,2
64,ADDRESS,328,CITY,353,STATE,363,ZIP,448,"TELEPHONE (   )   -
   EXT:"
9011 DATA 512,REFERRED BY,652,FIRST ACTIVE DATE,716,LAST ACTIVE
DATE,832,COMMENTS
```

Program Listing

See List of Advertisers on page 386

"The standard TRS-80 method of accepting input from its operator is about as efficient as punched card input."

- 1002 turns on the cursor in the first position of the input field.
- 1004 counts the number of characters input up to the maximum field length.
- 1005 accepts one character.
- 1010 is used if the character is a backspace. Go to the subroutine at 1050 (see below), then go to 1005.
- 1012 is used if the character is a forward arrow; turn it into a space and continue.
- 1015 is used if the character is the enter key, then the field is complete, so make the field length equal the maximum field length and then go to line 1030.
- 1019 is used if the character is neither alphabetic nor numeric; discard it and accept the next character by going to line 1005.
- 1020 appends the character to the previously accepted characters and displays it on the screen.
- 1030 repeats until the maximum field length is achieved, then turns off the cursor and returns to line 950.

Line 950 increments OE to point to the next element of array OM in preparation for the next call to this routine. It then returns to the calling program with variable OB$ containing the requested input string.

The subroutine at line 1050 is called from line 1010 if the input character is a backspace. Line 1055 handles a backspace from the first position of a field and prevents the cursor from moving into the title field. Lines 1060 and 1065 handle all other backspace conditions followed by a return to the input processing in the middle of line 1010.

Usage

In this example the calling program consists of lines one through 90 and provides no function other than data accumulation of a single screen. Rudimentary validation of the contents of OB$ between calls to the input array driver at line 900 is possible, but more extensive processing could cause the operator to overrun the input capabilities of the program. Any delays caused by input/output should be isolated to the time between screens or even better, after all input is complete. This, of course, will depend upon available storage.

Summary

The structure of these routines was chosen to facilitate their usage in as many programs as possible. This building-block approach will help to speed development of future programs. In keeping with this design technique, we have designated certain variables as belonging to common routines. The letters O and X are the initial letters of all variables within these routines. Additionally, although this listing does not demonstrate it, we have reserved variables starting with the letter Z for binary flags (yes/no).

Data collection is an important part of many applications. The standard TRS-80 method of accepting input from its operator is about as efficient as punched card input. These routines could speed up file building. ■

See List of Advertisers on page 386

A loop in a nest slows you in a rush.

Thoughts on For. . .Next

Richard Amyx
994 North Second St.
San Jose, CA 95112

A while ago I was tinkering with a little game I'd found in a book, the gist of which was to keep tabs on a nine-element array and to toss out array elements as certain conditions were met. The listing was relatively short and had no terribly complicated statements, so I keyed it in without paying much attention to what was going on within the program.

The routine was query/response, next query/response, and so on, with virtually no time lag between my responses and the next query—until the fifth or sixth response, at which point the program appeared to take off into outer space. After waiting a reasonable length of time I broke out of the program and proofread my keypunching around the break—no errors.

So I tried again and got the same result. But this time I decided to wait the machine out. When I eventually returned to the computer, I found that the program had indeed gone on as it should have. Why, I wondered, the sudden quantum leap in time between response and query?

I then looked at the program more closely where I'd broken out of it the first time, and found that the break had been in the middle of a sequence that looked like this:

```
10 FOR I = 1 TO 9
20 FOR J = 1 TO 9
30 FOR K = 1 TO 9
40 FOR L = 1 TO 9
.
.
.
100 NEXT L
110 NEXT K
120 NEXT J
130 NEXT I
140 GOTO xxx
```

It didn't take too much button-punching to learn that if this nest of For loops went all the way through it would require 7380 ($9 + 9^2 + 9^3 + 9^4$) total iterations. Combine that with a couple of If statements in the middle, and you can bet you're going to have to wait for a response.

Then looking at the program even more closely, I decided that it would require very little housekeeping to reduce the length of the array, and hence the upper limit of these For statements, as the game progressed. I wondered how much time could be saved if this one change were made. The givens were that the For loops would remain four deep and that the equation for the total number of iterations required was $Y = N + N^2 + N^3 + N^4$, where N was the upper limit of the For loop. A little more button-punching yielded the numbers in Table 1.

The numbers of iterations themselves ought to be impressive—certainly they're big enough—but what's even more important are the differences between them and the percentage of maximum time those differences represent. This is

Upper Limit	Total Iterations	% of Maximum
9	7380	100.0
8	4680	63.4
7	2800	37.9
6	1554	21.1
5	780	10.6
4	340	5.6
3	120	1.6
2	30	0.4

Table 1. Four-deep For loops.

✓See List of Advertisers on page 386

most strikingly illustrated by a graph (see Fig. 1).

As you can see, once you get past an upper limit of six you're going to be making some very powerful time trade-offs for every increment in the upper limit of the loops.

Getting back to the game, it appeared that the program usually wouldn't get into the heavy looping sequence until at least the third response, which meant that the upper limit could have been reduced from nine to seven and that a 62 percent saving in time might be realized. The next step, of course, was to test my hypothesis.

To do this, I wrote a short program just like the example sequence shown, filling in the dots with a couple of do-nothing If statements to increase overall execution time and thereby reduce the effect of my own reaction time on the stop watch. The results I got were (N again representing the upper limit of the loop index):

N = 9, t = 46 sec.
N = 8, t = 29
N = 7, t = 17

This was close enough to prove the theory. The execution of a very slow portion of the program could indeed be speeded up by at least 62 percent just by indexing the upper limits of the four For loops.

The rule is simple: The amount of time it takes to execute nested For...Next loops increases by the power of the depth to which the loops are nested—two-deep loops, time squared; three-deep loops, time cubed; and so on. Obviously, the best solution is not to nest For loops any deeper than absolutely necessary, but if you must nest them deeply, then pay very close attention to the ranges of the loop indices. ■

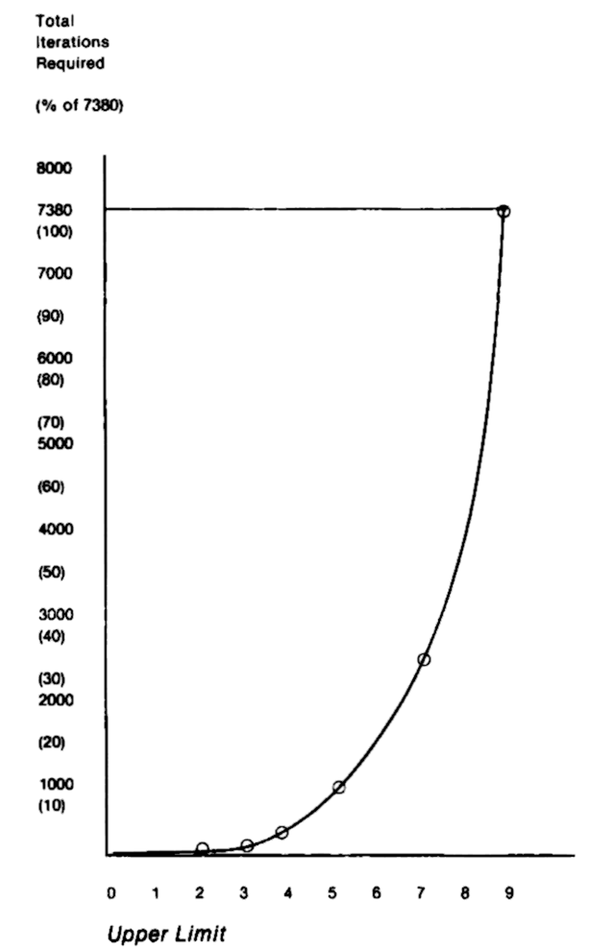


Fig. 1. Total iterations versus upper limit for four-deep For loops.

PROGRAMMING TOOLS FOR YOUR TRS-80™ MODEL I AND MODEL III

INSIDE LEVEL II

The Programmers Guide to the TRS-80 ROMS

INSIDE LEVEL II is a comprehensive reference guide to the Level II ROMs which allows the machine language or Basic programmer to easily utilize the sophisticated routines they contain. Concisely explains set-ups, calling sequences, and variable passage for number conversion, arithmetic operations, and mathematical functions, as well as keyboard, tape, and video routines. Part II presents an entirely new composite program structure which loads under the SYSTEM command and executes in both Basic and machine code with the speed and efficiency of a compiler. In addition, the 18 chapters include a large body of other information useful to the programmer including tape formats, RAM useage, relocation of Basic programs, USR call expansion, creating SYSTEM tapes of your own programs, interfacing of Basic variables directly with machine code, a method of greatly increasing the speed at which data elements are stored on tape, and special precautions for disk systems. INSIDE LEVEL II is a clearly organized reference manual. It is fully typeset and packed with nothing but useful information. It does not contain questions and answers, ROM dumps, or cartoons.
Includes updates for Model III. INSIDE LEVEL II.....$15.95

SINGLE STEP THROUGH RAM OR ROM

STEP80 allows you to step through any Basic or machine language program one instruction at a time, and see the address, hexadecimal value, Zilog mnemonic, register contents, and step count for each instruction. The top 14 lines of the video screen are left unaltered so that the target program may perform its display functions unobstructed. STEP80 will follow program flow right into the ROMs, and is an invaluable aid in learning how the ROM routines function. Commands include step (trace), disassemble, run in step mode at variable step rate, display or alter memory or CPU registers, jump to memory location, execute a CALL, set breakpoints in RAM or ROM, write SYSTEM tapes, and relocate to any page in RAM. The display may also be routed to your line printer through the device control block so custom print drivers are automatically supported.
Specify Model I or Model III. STEP80.....$16.95

TELECOMMUNICATIONS PROGRAM

This machine language program allows reliable high speed file transfers between two disk-based computers over modems or direct wire. It is menu driven and extremely simple to use. Functions include real-time terminal mode, save RAM buffer on disk, transmit disk file, receive binary files, examine and modify UART parameters, program 8 custom log-on messages, automatic 16-bit checksum verification of accurate transmission and reception, and many more user conveniences. Supports line printers and lowercase characters. With this program you will no longer need to convert machine language programs to ASCII for transmission, and you will know immediately if the transmission was accurate.
Specify Model I or Model III. TELCOM.....$39.95

PROGRAM INDEX FOR DISK BASIC

Assemble an alphabetized index of your entire program library from disk directories. Program names and free space are read automatically (need not be typed in) and may be alphabetized with a fast Shell Metzner sort by disk or program. The list may also be searched for any disk, program, or extension; disks or programs added or deleted; and the whole list or any part sent to the printer. Finally, the list itself may be stored on disk for future access and update. The best thing since sliced bread (January issue of '80 Microcomputing). Works with TRSDOS, NEWDOS, and NEWDOS 80. One drive and 32K required.
Model I only. Model III version soon. INDEX.....$19.95

4 SPEED OPTIONS FOR YOUR TRS-80

The SK-2 clock modification allows CPU speeds to be switched between normal, an increase of 50%, or a 50% reduction; selectable at any time without interrupting execution or crashing the program. Instructions are also given for a 100% increase to 3.54 MHz. The SK-2 may be configured by the user to change speed with a toggle switch or on software command. It will automatically return to normal speed any time a disk is active, requires no change to the operating system, and has provisions for adding an LED to indicate when the computer is not at normal speed. It mounts inside the keyboard unit with only 4 necessary connections for the switch option (switch not included), and is easily removed if the computer ever needs service. The SK-2 comes fully assembled with socketed IC's and illustrated instructions.
Model I only. SK-2.....$24.95

INSTANT ASSEMBLER

The INSTANT ASSEMBLER is a new, powerful tape-based assembler and debugger for the TRS-80. Now you can assemble directly to memory and immediately debug your program with the built in single stepping debugger. Quickly switch from assembler to debugger and back again without losing the source code. This feature makes INSTANT ASSEMBLER an excellent learning tool for assembly language programming.
INSTANT ASSEMBLER is absolutely unique among tape based assemblers in that it produces relocatable code modules that can be linked with the separate LINKING LOADER, which is supplied in two versions for loading programs into either high or low RAM. This lets you build long programs with small modules. INSTANT ASSEMBLER also features immediate detection of errors as the source code is entered, a compactly coded source format that uses 1/3 as much memory as standard source, and many operational features including single stroke entry of DEFB and DEFW, pinpoint control of listings, alphabetic listing of symbol table, separate commands for listing error lines or the symbol table, block move function, and verification of source tapes.
INSTANT ASSEMBLER's debugger provides single stepping with full register displays, decimal or hex entry of addresses, forward or backward memory displays, disassembly of object code in memory, memory display in ASCII format, and hex-to-decimal or decimal-to-hex conversion. The single-stepper will step one instruction at a time or at a fast rate to any defined address.
INSTANT ASSEMBLER occupies less than 8400 bytes of memory. In a 16K machine this will leave you enough memory to write assembly language programs of around 2000 bytes. This and its module-linking feature make INSTANT ASSEMBLER ideal for users with only 16K machines. The instruction manual may be purchased separately for $5, which will apply towards the purchase of the INSTANT ASSEMBLER.
Specify Model I or Model III. INTASM.....$29.95

RAM SPOOLER AND PRINT FORMATTER

This program is a full feature print formatting package featuring user defineable line and page length (with line feeds inserted between words or after punctuation), screen dump, printer pause control, and baud rate selection. In addition, printing is done from a 4K expandable buffer area so that the LPRINT or LLIST command returns control to the user while printing is being done. Ideal for Selectric or other slow printers. Allows printing and processing to run concurrently. Output may be directed to either the parallel port, serial port, or the video screen.
Specify Model I or Model III. SPOOLER.....$16.95

MACHINE CODE FAST FOURIER TRANSFORM

This complete package includes 3 versions of the machine language FFTASM routine assembled for 16, 32, and 48K machines, a short sample Basic program to access them, a 10K Basic program which includes sophisticated interactive graphing and data manipulation, and a manual of instructions and examples. The machine language subroutines use variables defined by a supporting Basic program to make data entry and retrieval extremely fast and easy for custom implementation. They perform 20 to 40 times faster than their Basic equivalent (256 points in 12.5 seconds), and require less than 1550 bytes of memory. The FFT is useful in analyzing stock market and comodity trends as well as for scientific information.
Specify Model I or Model III. FFTASM.....$49.95

DUPLICATE SYSTEM TAPES WITH CLONE

Make duplicate copies of any tape written for Level II. They may be SYSTEM tapes or data lists. The file name, load address, entry point, and every byte (in ASCII format) are displayed on the video screen. Model III version allows changing tape speed.
Specify Model I or Model III. CLONE.....$16.95

RAMTEST FOR LEVEL II

This machine language program is a very thorough test for several types of RAM errors. A complete test of each individual bit in a 48K machine takes just 14 seconds. Includes a separate test for power line glitches.
Model I only. RAMTEST.....$9.95

EDIT BASIC PROGRAMS WITH ELECTRIC PENCIL

Load Basic programs or any other ASCII data file into the disk version of Electric Pencil for editting. One command from DOS quickly modifies existing files to Pencil format. One disk and 32K required.
Model I only. PENPATCH.....$9.95

MUMFORD MICRO SYSTEMS ✓144

ORDERING: Complete satisfaction is guaranteed or a full refund will be made. All programs are shipped on cassette unless $5 is included for a 35 track formatted (no system) disk. Include $1.50 for postage and handling. California residents add 6% sales tax. Visa, Mastercharge and COD orders accepted. SPECIFY MODEL I OR MODEL III. Dealer inquiries invited.

Box 400-E Summerland, California 93067 (805) 969-4557

✓See List of Advertisers on page 386

Can you say "data?"

Datafix

A. J. Barnard, Jr.
J.T. Baker Chemical Company
Phillipsburg, NJ 08865

A. D. Barnard
Rensselaer Polytechnic Institute
Troy, NY 12181

The TRS-80 keypad can help you whenever you're entering programs with long data statements—for example, music and graphics programs, science programs or long data lines to be POKEd into machine-code routines.

The keypad, you say, only allows you to use ten digits, a period (decimal point) and the enter key. You'll still have to enter the word data, any necessary commas and minus signs from the keyboard.

Not necessarily. With the following routine, you can translate two or more *consecutive* periods into the word data, a comma, or a minus sign. Datafix is a Basic routine for a Model I Level II TRS-80 that gives you the proper syntax from the keypad, using only four variables.

```
0 GOTO10
1 ZX=PEEK(16548)+256*PEEK(16549)
2 IF(PEEK(ZX)+PEEK(ZX+1)=0)THEN6
  ELSEIFPEEK(ZX+4)<>46THEN5
  ELSEZU=0:ZV=0:ZW=0
  :FORI=ZX+4TOZX+255:IFPEEK(I)=0THEN5
  ELSEIF(I=ZX+5ANDPEEK(I-1)=46ANDPEEK(I)=46)
  THENPOKEI-1,136:POKEI,32:ZU=I:NEXT
3 IF(ZV=0ANDPEEK(I)=34)THENZV=1:NEXT
  ELSEIF(ZV=1ANDPEEK(I)<>34)THENNEXT
  ELSEIF(ZV=1ANDPEEK(I)=34)THENZV=0:NEXT
4 IF(ZV=0ANDPEEK(I-1)=46ANDPEEK(I)=46)
  THENPOKEI-1,44:POKEI,32:ZW=I:NEXT
  ELSEIF(I=ZU+1ORI=ZW+1)ANDPEEK(I)=46
  THENPOKEI,45:ZU=0:ZW=0:NEXT
  ELSENEXT
5 ZX=PEEK(ZX)+256*PEEK(ZX+1):GOTO2
6 PRINT,"DATA STATEMENTS CORECTED.":END
10 REM * MAIN PROGRAM STARTS HERE *
```

Program Listing 1. Datafix Routine

Changing Code

If you enter two consecutive periods after a line number, Datafix recognizes that the line is a data statement and, by POKEs, changes the first period to the word data (internal code 136) and the second to a space (32).

At any further point in the statement, if you enter two consecutive periods, the routine changes them to a comma (44) and a space (32). If you enter three consecutive commas, the third is converted to a minus sign (45). The routine ignores any periods within quotation marks and leaves a space after the word data and between each item.

For example, you can enter the following lines, except for alphabetic and special characters, using the keypad only:

```
100..10.1..40.2
105...10.1...40.2
110..JONES..40.2
115.."BYE NOW..."...10.1
```

Upon enterning: RUN 1, Datafix converts these statements to:

```
100 DATA 10.1, 40.2
105 DATA -10.1, -40.2
110 DATA JONES, 40.2
115 DATA "BYE NOW...", -10.1
```

and then advise you that the data statements were corrected.

As written, you can load Datafix before your program. You can also renumber the routine to, say, 60001–60006 and append it to a program in memory (see, for example, *80 Microcomputing*, January 1981, p. 213 for the PEEK/POKE approach).

If you are entering and editing statements only once, you can delete the routine after you use it.

How It Works

In Program Listing 1, we use line feeds and indents to make the logic clearer. Dropping this nicety, reduces memory requirements. Line 0 bridges the routine and returns to the main program at line 10.

Line 1 returns the decimal address, ZX, of the first line of a Basic program.

Line 2 first checks the two bytes of the line pointer; if they equal zero, the routine goes to line 6 and ends. Line 2 also checks for a period immediately following the line number; if none exists, the line is not a data statement and the routine passes to line 5.

Finally, line 2 can also start a character-by-character PEEK of a statement to be edited. When it meets character zero, Datafix has reached the end of the line, and passes to line 5.

If line 2 finds that the first two characters of a program line are periods, the periods are changed to the codes for the word data and a space. ZU is set to the character position I.

Line 5 finds the address of the next program line and returns to line 2.

Line 3 checks for a quotation mark (code 34). If one is found, flag ZV is set and Datafix will not check for periods until either a second quotation is encountered or the line ends.

If Datafix finds two consecutive periods beyond the start of the data statement, but not within quotation marks, line 4 changes them to a comma and a space, and ZW is set to the character position I.

Then, if the routine meets a third period and confirms it as consecutive by I *being equal to* ZU + 1 or ZW + 1, it is converted to a minus sign. ■

Strengthen NEWDOS 80's chain command.

Weak Link

Steven Denholtz, D.M.D.
Livingston Professional Building
114 West Mt. Pleasant Ave.
Livingston, NJ 07039

Unlike its use under Newdos80, the Chain function usually exits and erases a resident Basic program, then inputs another Basic program from disk. This program, in turn, operates on a still resident set of variables and/or files that are shared by related programs. The rationale is to reduce the amount of user memory the program consumes so a larger share of memory can be used for data or variables.

In Newdos80, however, the Chain function performs an entirely different operation. A short Basic utility is provided here so you can use Chain with speed and ease not available under Newdos80.

What is Chain?

Chain is a DOS library command. After a Chain command, anytime a program asks for input from the keyboard, the keyboard record will automatically be supplied from a disk chain file without any operator action. How did the keyboard answers get on the disk? How does the computer know where to look? To answer these questions, let's learn a few new terms and look at what a chain file is:

```
Chain File:  Section0
             Section1
             Section2
             ....>etc. til EOF
```

Syntax

Each section is an independent subfile of keyboard responses. Multiple sections can share the same file name. This avoids wasting a whole granule of disk space (1.25k) for each group of keyboard responses that usually takes a few words (100 bytes maybe). Thus the chain file was designed to have sections that are really independent subfiles sharing one filespec. This saves valuable disk space. You address the section when you issue the Chain command:

2a. 'CHAIN filespec <SECTIONn>' if in Dos; or,
2b. 'CMD''CHAIN filespec <SECTIONn>'' ' if in Disk Basic.

SECTIONn is optional on the first section (0). If the chain file contains more than one section, you must specify which one you are addressing, where n is a positive integer. If SECTIONn is deleted when issuing the Chain command, the default naturally is the first section, SECTION0. The filespec in Fig. 2 has a default extension of JCL, (which stands for Job Control Language, but that is getting beyond the scope of this article).

```
Section:  <special character>
          <SECTIONn>
          keyboard record
          End Of Line character (OD H or
          Enter)
```

The special character is always one of the following:

CHR$(128) (80 H) : start of a new Section
CHR$(129) (81 H) : display message, await ENTER before proceeding
CHR$(130) (82 H) : chain file "remark"
CHR$(131) (83 H) : display message, no pause

The special character 128 (80 Hex) is mandatory when starting a new section, except for the first section. All other characters are strictly optional. But, after using a CHR$(128) to start a new section, you must also put SECTIONn immediately after it so you can later address which subfile (or section) you wish to use when you invoke Chain, as shown in Fig. 2.

Uses of Chain

The keyboard record is the answer that will be displayed on the screen and used as if it were actually input from the keyboard (see Fig. 3). This can supply a commonly used set of parameters to an often used utility. More often, it may be a series of System and Basic commands implementing a complex program such as shown in Fig. 5.

My diskettes include a "JCL/JCL" file to be used in an Auto boot mode—every diskette, that is, but my Newdos80 system diskette. That one has the Auto command: AUTO CHAIN JCL/JCL, but has no file named JCL/JCL. This way, whenever I use my computer, I simply place either the system diskette or my application

```
LCDRIVER
BASIC
LOAD"LEDGERB"
CLS
LIST1
LIST10030-10040
EDIT10040

DOS READY
CHAIN JCL/JCL:1
DOS READY
LCDRIVER
GENERAL PURPOSE LOWER CASE DRIVER
DOS READY
BASIC
DISK BASIC. RADIO SHACK'S ROM ENCHANCED WITH APPARAT'S NEWDOS80
EXTENDED AND DISK FEATURES.
```

Fig. 5 A Typical Use of Chain.

See List of Advertisers on page 386

diskette in drive zero and hit reset. The chain file issues all commands necessary to place my computer in a state where I can sit down and set to work. This may include listing any particular program lines that I feel need further development, or placing chain file messages to remind me where I left off, and even entering the Edit mode already pointing to a troublesome line.

Last Minute Hints

The operator may terminate the Chain command any time by pressing the up-arrow; the operator may force a Chain pause by holding the right-arrow, and continue by pressing [enter].

An Original Utility to Create Chain Files

Program Listing 1, Makejcl/bas, creates and appends chain files with every feature described so far fully implemented. Study the program with its remarks. Type it into your system and save it on your Newdos80 system diskette. You will undoubtedly use it often and perhaps come up with your own applications. ■

```
1 '

*****************************************************************
*    "MAKEJCL/BAS" - BY STEVEN DENHOLTZ, B.S., D.M.D.           *
*           COPYRIGHT 1980    ALL RIGHTS RESERVED                *
*****************************************************************


5 CLS:CLEAR400:PRINT      "MAKEJCL/BAS -  Will create a  j.c.l.
file to be used by CHAIN
                     by Steven Denholtz, D.M.D.  (201) 347-1277"

10 D$="xxxxxxxxxxxxxxxxxxxxxxxxxxx":D1$="xxxxxxxxxxxxxxxxxxx"
30 PRINT"
":LINEINPUT"Filespec (NO error checking ! MUST have '/jcl' exten
sion)
       :?";FILE$
40 INPUT"Opening (O) a new file ,Appending last record (A) or
new SECTIONid (N) ";D$
45 IF D$="N"OR D$="n" THEN 300
47 IF D$="O" OR D$="o" THEN 400
50 OPEN"E",1,FILE$ :' open to append to existing record.
100 PRINT@640,CHR$(255)+CHR$(255):PRINT@640,""
105 GOSUB 500:PRINT@640,"";:LINEINPUT"DOS or DISK BASIC Command
? ";D$
110 GOSUB600 :' Interpret first character for special meaning
120 PRINT#1,D$ :' Put it into disk Chain file.
150 PRINT@704,"";:LINEINPUT"Enter another command (Y/N) ";D1$
160 IF D1$="y" OR D1$="Y" THEN 100 ELSE 200
200 CLOSE
210 PRINT"   *** END ***"
220 END
300 'MUST LOOK UP LAST SECTIONID, ADD ONE TO IT THEN GOTO350
310 OPEN"I",1,FILE$
315 INPUT#1,D$
320 IFASC(D$)=128 THEN SECTINID =SECTINID+1
325 IF NOT EOF (1) THEN 315
330 SECTINID=SECTINID+1:PRINT@170,"******* SectionID=";SECTINID
340 CLOSE
350 OPEN"E",1,FILE$ :' open for the purposes of appending a new
Section to an existing file
360 D1$=CHR$(128)+"SECTION"+RIGHT$(STR$(SECTINID),1)+CHR$(13)
370 PRINT@640,CHR$(255)+CHR$(255):PRINT@640,""
375 GOSUB500:PRINT@640,"";:LINEINPUT"DOS or DISK BASIC Command ?
";D$
377 GOSUB600 :' interpret first character for special meaning
380 PRINT#1,D1$+D$ :'Put it into the disk Chain file.
395 GOTO150
399 END
400 OPEN"O",1,FILE$ :' open a new file (this will erase an exist
ing file with the same filespec !)
420 GOTO100 :' pretty up the screen
500 FORX%=736TO992STEP64:PRINT@X%,".";:NEXT:PRINT@739,"(,) user
comment with pause";:PRINT@803,"(.) user comment";:PRINT@867,"(/
) file comment";
550 RETURN
600 ' "Interpret first character of Command
610 IF LEFT$(D$,1)="," THEN D$=CHR$(129)+D$
620 IF LEFT$(D$,1)="." THEN D$=CHR$(131)+D$
630 IF LEFT$(D$,1)="/" THEN D$=CHR$(130)+D$
640 RETURN
```

Program Listing 1

Your best Model I/III peripheral buy is a modem.

Your best Model I/III modem buy is LYNX.

Considering expanding your TRS-80 Model I or III? Then you should know that **only one kind of peripheral** can give you all these **extra capabilities** . . . in a single unit:

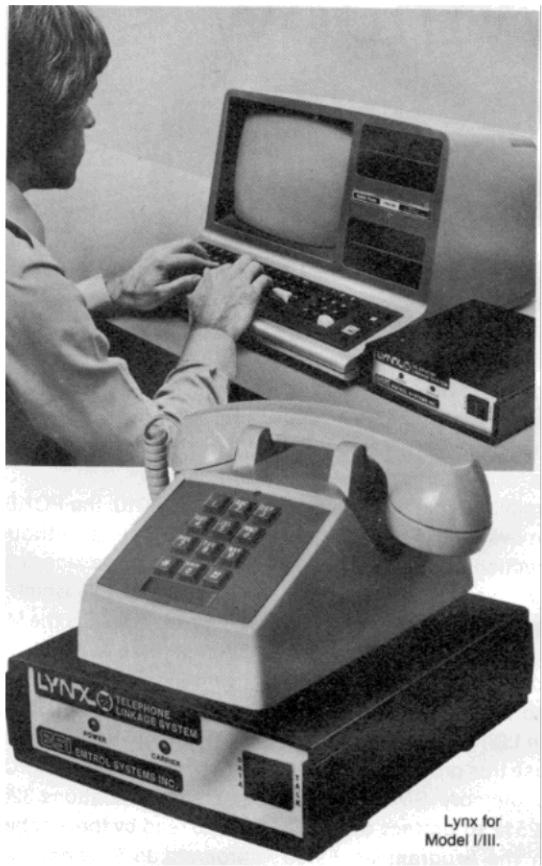
- **Computer-to-computer data communications, for better personal or business management.**
- **Access to free "bulletin board" resources.**
- **Access to *Source*, *CompuServe* and other time-sharing systems.**
- **Services like electronic mail, instant news and financial reports.**
- **An almost limitless supply of new games and other enthusiast activities.**

Of course, we're talking about a modem. A far more **sensible** first step in expanding than, say, joysticks, or a voice box, or a printer. And, therefore, your **best peripheral buy.**

A modem opens up a whole new world to you. A world of **thousands of computers** and computer people just like you, in homes and businesses around the block and across the country. A world you tap through your telephone.

But don't simply settle for any modem. For the most modem satisfaction, you need to make the **best modem buy.**

And that's **LYNX.**



Lynx for Model I/III.

The new LYNX for TRS-80 Model I and Model III. It's the latest innovation from the people who are making data communications affordable. It's the best first step you can take in expanding. It makes your microcomputer a whole new animal.

The new LYNX is the ideal modem for your Model I or III. That's because it's the **key element** of a **total direct-connect telephone communications package.**

This package includes all serial and computer bus interfaces, cables and terminal software . . . and comes to you with these remarkable features:

- **SINGLE PRICE: $299.95!**
- **Compatible with both Model I and III.**
- **No separate purchases required.**
- **One-year warranty.**
- **Auto dial/auto answer.**
- **Works with any software, including ST-80 by Micklus.**
- **Active clear, break, and "control" keys.**
- **Half or full duplex, variable word length, parity and stop bits.**
- **Dial from phone, keyboard, or memory.**
- **Works with or without Model I expansion interface.**
- **Works with or without Model III RS-232 card.**
- **Can be placed on either side of Model III.**
- **No tools needed to install.**

No other modem gives you so much in a **single package.** And when you **compare** the LYNX price with the **total cost** of bringing any other modem on line, it could **save you OVER $100!**

ESI LYNX

✓278

123 LOCUST STREET LANCASTER, PENNSYLVANIA 17602
Phone 717/291-1116

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation.
FCC Registration Number: A909KE-68171-DM-N

Consult your local microcomputer retailer, or call one of these LYNX handlers today—TOLL FREE:

ADVENTURE INTERNATIONAL
Longwood, FL
800/327-7172

ADVANCED COMPUTER PRODUCTS
Irvine, CA
800/854-8241

TSE HARDSIDE
Milford, NH
800/258-1790

COMPUTER PLUS
Littleton, MA
800/343-8124

THE PROGRAM STORE
Washington, DC
800/424-2738

STEVENS RADIO SHACK
Phoenixville, PA
800/345-6279

SIMUTEK
Tucson, AZ
800/528-1149

PROGRAMS UNLIMITED
Jericho, NY
800/645-6038

✓See List of Advertisers on page 386

A machine-code subroutine to dump memory on your display.

Memory Window

Warren Merkey
824 S.E. 1st Avenue
Gainesville, FL 32601

Now you can view any 1024-byte block of memory in your Model I Level II keyboard—all 32 K. Memory window dumps memory onto the screen using a machine-code subroutine (a USR).

Here is how the program works. Line 1010 tells the computer where the USR is located and also makes all variables into integers. Line 1020 is the USR in the form of decimal data. Line 1030 loads the USR into place in high memory. Line 1040 asks the user for a place to start. Line 1050 calls the subroutine that moves memory onto the screen. Lines 1060–1080 are a loop that looks for control key input (arrow keys, Enter, and space bar). Line 2000 prints the current value of M in the upper left corner of the screen, with an arrow pointing to M. Line 9000 keeps the value of M within the first 32K of memory. Line 9010 calculates decimal POKE values. (This will be explained below.) Line 9020 POKEs the address of the memory block to be displayed into the USR and then calls the USR.

The USR consists of 12 bytes of code, in which the BC, DE and HL register pairs are loaded with the block-move parameters, which is done by the LDIR instruction. As the display scrolls the HL pair is the only one changed but the BC and DE pairs can also be easily changed. With these 12 bytes of code and some skillful POKEing, you can move any part of memory to any other location. By POKEing a 184 where there is a 176 (BO hex), you change the LDIR to LDDR. This is sometimes necessary when moving a block of memory into an overlapping area, where the direction of the move would cause memory to be overwritten before it is moved. (Refer to "Now You See It" in *80 Microcomputing*, February 1981 for another application of the 12-byte LDIR subroutine.)

To use this program, respond to the Memory Size question with 32512 to protect the USR. Key in the program as listed and save it on tape before running it. The program may bomb when it scans through certain sensitive areas of memory. The four arrow keys give you scrolling in the four directions. The enter key allows you to stop and pick another starting place. The space bar causes the current memory location (and ASCII value) to be printed in the upper left corner.

In line 9010, the POKE values must be decimal, although they are related to hexadecimal. Take the value 32600 which equals 7F58 hex. The 7F is the Most Significant Byte and the 58 is the Least Significant Byte. The LSB and MSB converted back to decimal are POKE values 88 and 127. These values are POKEd into memory locations 32519 and 32520 read by the machine-code program as 7F58 hex.

Note that a hex number is translated to decimal by multiplying the values of its digits. 7F58 hex translates as $7 \times 4096 + F \times 256 + 5 \times 16 + 8 \times 1$ equals 326000. Remember that F equals 15. To reverse the procedure, you begin by dividing by 4096. The first two statements of line 9010 are long division:

```
      7 = I       (I = INT(M/4096))
4096)32600
     28672
      3928 = J.   (J = M - I*4096)
```

The third and fourth statements are long division by 256:

```
     15 = K       (K = INT(J/256))
256)3928
    3840
      88 = L.     (L = J - K*256)
```

The next divisor would be 16; however, as you can see, 88 is the decimal POKE value we need for the LSB. This leaves us with a seven and a 15, which we know is 7F hex, a one-byte number. If we multiply seven by 16 and add 15, we get 127, the decimal equivalent of 7F hex. This last step is expressed as $H = K + 16*I$. Lines 9000–9020 are executed every time you press an arrow key. The USR reloads the entire screen with memory in the blink of an eye, giving the illusion of scrolling.

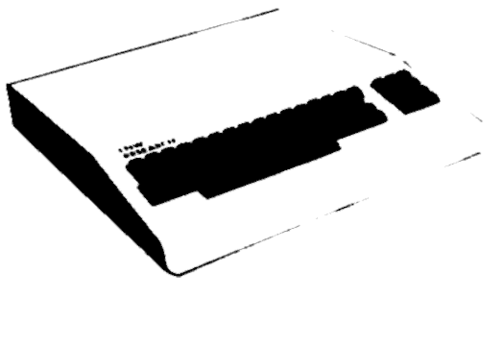
I have used the algorithms in this program to build a memory editor which works reasonably well as a word processor. At the very least, Memory Window may impart some knowledge to those of you who can grasp its principles. ■

```
1000 'MEMORY WINDOW
1010 POKE16526,0:POKE16527,127:DEFINTA-Z
1020 DATA 1,0,4,17,0,60,33,0,0,237,176,201
1030 A=32512:FORB=0TO11:READC:POKEA+B,C:NEXTB
1040 CLS:INPUT"STARTING POINT (0 - 31743)";M
1050 GOSUB9000
1060 C=PEEK(14400):IFC=0THEN1060ELSEIFC=8THENM=M-64
1070 IFC=16THENM=M+64ELSEIFC=32THENM=M-1ELSEIFC=64THENM=M+1
1080 IFC=1THEN1040ELSEIFC=128THEN2000ELSE1050
2000 PRINT@1,CHR$(93);"-";M;PEEK(M);:GOTO1060
9000 IFM<0THENM=0ELSEIFM>31743THENM=31743
9010 I=INT(M/4096):J=M-I*4096:K=INT(J/256):H=K+16*I:L=J-K*256
9020 POKE32519,L:POKE32520,H:X=USR(0):RETURN

USR ASSEMBLY LISTING

7F00            00100  ORG   32512     ;USR ADDRESS
7F00  01 00 04  00110  LD    BC,1024   ;LENGTH OF BLOCK
7F03  11 00 3C  00120  LD    DE,15360  ;START OF SCREEN MEMORY
7F06  21 00 00  00130  LD    HL,0      ;POKEd FROM BASIC PROGRAM
7F09  ED B0     00140  LDIR            ;BLOCK-MOVE INSTRUCTION
7F0B  C9        00150  RET             ;RETURN TO BASIC
7F00            00160  END             ;END ASSEMBLY PROGRAM
```

Program Listing

★ ★

4MHZ, DOUBLE DENSITY, COLOR & B/W GRAPHICS . . THE LNW80 COMPUTER



When you've compared the features of an LNW80 Computer, you'll quickly understand why the LNW80 is the ultimate TRS80 software compatible system. LNW RESEARCH offers the most complete microcomputer system at an outstanding low price.
We back up our product with an unconventional 6 month warranty and a 10 days full refund policy, less shipping charges.

LNW80 Computer $1,450.00
LNW80 Computer w/B&W Monitor & one 5" Drive $1,914.00
All orders must be prepaid, CA residents please include 6% sales tax.
Contact us for shipping charges

* TRS80 Product of Tandy Corporation.
** PMC Product of Personal Microcomputer, Inc.

COMPARE THE FEATURES AND PERFORMANCE

FEATURES	LNW80	PMC-80**	TRS-80* MODEL III
PROCESSOR	4.0 MHZ	1.8 MHZ	2.0 MHZ
LEVEL II BASIC INTERP.	YES	YES	LEVEL III BASIC
TRS80 MODEL 1 LEVEL II COMPATIBLE	YES	YES	NO
48K BYTES RAM	YES	YES	YES
CASSETTE BAUD RATE	500/1000	500	500/1500
FLOPPY DISK CONTROLLER	SINGLE/DOUBLE	SINGLE	SINGLE/DOUBLE
SERIAL RS232 PORT	YES	YES	YES
PRINTER PORT	YES	YES	YES
REAL TIME CLOCK	YES	YES	YES
24 X 80 CHARACTERS	YES	NO	NO
VIDEO MONITOR	YES	YES	YES
UPPER AND LOWER CASE	YES	OPTIONAL	YES
REVERSE VIDEO	YES	NO	NO
KEYBOARD	63 KEY	53 KEY	53 KEY
NUMERIC KEY PAD	YES	NO	YES
B/W GRAPHICS, 128 X 48	YES	YES	YES
HI-RESOLUTION B/W GRAPHICS, 480 X 192	YES	NO	NO
HI-RESOLUTION COLOR GRAPHICS (NTSC), 128 X 192 IN 8 COLORS	YES	NO	NO
HI-RESOLUTION COLOR GRAPHICS (RGB), 384 X 192 IN 8 COLORS	OPTIONAL	NO	NO
WARRANTY	6 MONTHS	90 DAYS	90 DAYS
TOTAL SYSTEM PRICE	$1,914.00	$1,840.00	$2,187.00
LESS MONITOR AND DISK DRIVE	$1,450.00	$1,375.00	---

LNW80

- BARE PRINTED CIRCUIT BOARD & MANUAL $89.95

The LNW80 - A high-speed color computer totally compatible with the TRS-80*. The LNW80 gives you the edge in satisfying your computation needs in business, scientific and personal computation. With performance of 4 MHz, Z80A CPU, you'll achieve performance of over twice the processing speed of a TRS-80*. This means you'll get the performance that is comparable to the most expensive microcomputer with the compatibility to the world's most popular computer (TRS-80*) resulting in the widest software base.

FEATURES:
- TRS-80 Model 1 Level II Software Compatible
- High Resolution Graphics
  - RGB Output - 384 x 192 in 8 Colors
  - NTSC Video or RF MOD - 128 x 192 in 8 Colors
  - Black and White - 480 x 192
- 4 MHz CPU
- 500/1000 Baud Cassette
- Upper and Lower Case
- 16K Bytes RAM, 12K Bytes ROM
- Solder Masked and Silkscreened

LNW SYSTEM EXPANSION

- BARE PRINTED CIRCUIT BOARD AND MANUAL $69.95
- WITH GOLD CONNECTORS $84.95

The System Expansion will allow you to expand your LNW80, TRS-80*, or PMC-80** to a complete computer system that is still totally software compatible with the TRS-80* Model 1 Level II.

FEATURES:
- 32K Bytes Memory
- 5" Floppy Controller
- Serial RS232 20ma I/O
- Parallel Printer
- Real Time Clock
- Screen Printer Bus
- On Board Power Supply
- Solder Masked and Silkscreened

LNDoubler & DOS PLUS 3.3D

- Assembled and Tested W/DOS PLUS 3.3D. $175.00

Double-density disk storage for the LNW Research's "System Expansion" or the Tandy's "Expansion Interface". The LNDoubler™ is totally software compatible with any double density software generated for the Percom's Doubler***. The LNDoubler™ provides the following outstanding features.

- Store up to 350K bytes on a single 5" disk
- Single and double density data separation
- Precision write precompensation circuit
- Software switch between single and double density
- Easy plug in installation requiring no etch cuts, jumpers or soldering
- 35, 40, 77, 80 track 5" disk operation
- 120 day parts and labor Warranty

*** Doubler is a product of Percom Data Company, Inc.

DOS PLUS 3.3D

Micro Systems software's double density disk operating system. This operating system contains all the outstanding features of a well developed DOS, with ease in useability.

KEYBOARD

LNW80 KEYBOARD KIT $84.95

The Keyboard Kit contains a 63 key plus a 10 key, P.C. board, and remaining components.

CASE

LNW80 CASE $84.95

The streamline design of this metal case will house the LNW80, LWN System Expansion, LNW80 Keyboard, power supply and fan, LNDoubler™, or LNW Data Separator. This kit includes all the hardware to mount all of the above. Add $12.00 for shipping

PARTS AVAILABLE FROM LNW RESEARCH
- 4116 - 200ns RAM
  - 6 chip set $26.00
  - 8 chip set $33.50
  - 16 chip set $64.00
  - 24 chip set $94.00
  - 32 chip set $124.00

- LNW80 "Start up parts set" LNW80-1 . . . . $82.00
- LNW80 "Video parts set" LNW80-2 . . . . $31.00
- LNW80 Transformer LNW80-3 . . . . $18.00
- LNW80 Keyboard cable LNW80-4 . . . . $16.00
- 40 Pin computer to expansion cable $15.00
- System Expansion Transformer $19.00
- Floppy Controller (FD1771) and UART (TR1602) . . . $30.00

LNW RESEARCH

C O R P O R A T I O N

2620 WALNUT

TUSTIN CA. 92680 ✓571

ORDERS & INFO. NO. 714-544-5744

SERVICE NO. 714-641-8850

VISA MASTER CHARGE ACCEPTED

UNLESS NOTED ADD $3 FOR SHIPPING

●●

RELOAD 80

Many of our readers have called with questions about Color Computer programs and LOAD80. Some wanted to know if Color Computer programs were on the tapes, others wondered if they could load the LOAD80 programs into their Color Computer, alter them and run them. This month, we'll clear the air about the Color Computer.

First, the Color Computer programs we publish in the magazine are *not* on the LOAD80 tapes. This is because the Color Computer downloads data at 1500 baud *only*. LOAD80 tapes are recorded at 500 baud so they can be used by both Model I and Model III users. LOAD80 tapes with programs recorded at different baud rates cannot be duplicated by our high speed duplicator.

You cannot load the LOAD80 tapes into the Color Computer for two reasons: one, the baud rates are different, and two, the Color Computer uses two separate tone pulses to output data—one to indicate a binary 1, the other to indicate a binary 0. The Models I and III use a single tone pulse that is either on or off to indicate the respective binary 1 or 0. For these reasons, the Color Computer and LOAD80 are incompatible.

Of course, you can still modify the programs in *80 Microcomputing* to comply with Color Basic, and then type in the listings.

At the moment, there are plans afoot to put together a collection of the magazine's Color Computer programs in a special edition, Color LOAD80. This special edition should be available by Christmas. Check this column next month for full details.

Model II owners have been asking for a LOAD80 of Model II programs, as well. This project is also under development and an announcement is planned for our December issue. The Model II LOAD80 will be a bit more expensive than either the regular LOAD80 or the special Color LOAD80, since it will be disk based.

The response to the LOAD80 project has been almost overwhelming. The Editors of the magazine are glad to be able to make this service available. We appreciate your feedback. Send your LOAD80 comments to this column, care of *80 Microcomputing*, 80 Pine Street, Peterborough, NH 03458.

Next month, we'll delve into the mysteries of the Editor/Assembler, a subject postponed from this month because of the great number of Color Computer inquiries. ■

October LOAD80 Directory

PGM#	Filename	Page	Comments
1	MASTMIND	122	None
2	SYLLOG	132	None
3	VOICE	141	None
4	SPEAK	141	None
5	TYPE	141	None
6	TALKER	146	None
7	BSKTBALL	184	None
8	HOMEINVT	220	None
9	BOOKS	234	None
10	SUNSET	272	None
11	FREEZER	274	None
12	TRSCPM	288	None
13	YAHTZEE	302	None
14	BRIDGE	306	None
15	FIREFGHT	316	None
16	SPACE	334	None
17	JERICHO	350	None
18	TPRGEN/SCR	196	Needs EDTASM
19	AUTOKEY/SRC	242	Needs EDTASM
20	ASMPTCH/SRC	326	Needs EDTASM
21	EDASFIX/SRC	344	Needs EDTASM
22	COPYIT/SRC	370	Needs EDTASM

Do You Have Disk Errors?

Over 90% of Disk errors are caused by two things. Bad disks and not having a data seperator. If you haven't ordered Percom's Doubler II then see our ad for details and order one now. You will not believe the improvement in your system. If you are having CRC errors, Data not found, or other errors especially in the higher numbered tracks you probably have a bad or worn out disk.

Every disk sold is certified but there is a big difference between brands. Some brands just barely pass the ANSI certification standards, others like WABASH produce disks that are far better than the minimum standards. (Wabash sells the disks that aren't good enough to put the Wabash Label on to Discount outlets that sell them as off brands and no Label disks.) WABASH Disks are certified over the entire useable disk surface. Despite repeated use and normal wear and calibration drift on your drive system, WABASH disks will continue to work reliably - long after the so called bargain brands have failed. That's why WABASH guarantees all their Disks for a minimum of 2 years.

Percom ran extensive tests during their research and design of the doubler and reached the conclusion that Wabash made the best 5¼ inch double density diskette.

Order some today if you don't like them for any reason return them within 30 days and we will give you a prompt refund. If you have any problems with them in the next two years return the disk and we will send you a new one. No other company can match that offer at our LOW prices.

Single Density 35/40 tracks $25.95
Double Density 35/40 tracks $32.99
Double Density 77/80 tracks $32.99
Double Density 80 tracks 2 sided 39.00
8 Inch Floppies, Double Density $39.00
SAVE MORE, ORDER TWO OR MORE BOXES AND TAKE A DOLLAR A BOX DISCOUNT.

FREE PRINT BUFFER!

What's a print bufffer you ask? It is hard to describe physically it's a 8x10x4 box that has 16k memory and plugs in between the computer and the printer. The printer can only accept data a line at a time with the Epson mx-80 thats around 100 baud. The computer can send print data through the printer port at over 1000 baud. The print buffer takes the data from the computer and stores it until the printer is ready for it.

If you are using your computer for business or if your time is valuable and you hate to sit doing nothing while the printer is printing YOU NEED THIS. With the time, the buffer saves you, it will pay for itself within a few days. We are so sure that you will like it that you can use it for 30 days free. If you don't think it will pay for itself send it back for a prompt refund.

MICROCOMPATIBLE with 16K of Memory $290.00
Cable .... $ 35.00

TWO NEW ARCADE GAMES

DUNGEON ESCAPE

It's been on the market for two months and has been getting rave reviews. Some of the comments are "I like it better than all of the previous dungeon games. It's got the great graphics and it's fast" "I get tired of the Big Five™ arcade games as they are the same every time. Dungeon Escape gives you a different dungeon and a different ghost every game." "My favorite game." "I used to fall asleep waiting for the graphics in Hellfire Warrior™, Dungeon Escape is 10 times as fast".

SUPER GRAPHICS ★ INTERACTIVE SOUND
LATEST PROGRAMMING TECHNIQUES

SCARFMAN

A new game by the author of Adventure Internationals Missle Attack. This is very similar to the PAC-MAN arcade game.
It has excellent graphics, with sound, and you can even use the joy sticks with this one.
YOU MUST SPECIFY MODEL I OR MODEL III.
Available on:
16K Tape $15.95
Diskette, 32K $19.50

PERCOM'S DOUBLER II

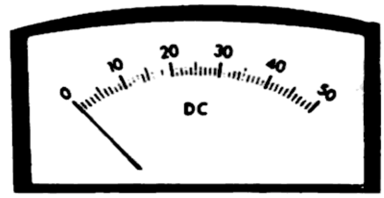
If you haven't got a doubler for your Model I, order one now. this is the greatest thing to come along for the Model I. Since it actually reduces disk read/write errors, your system will run better with a PERCOM doubler. And you will also save both time and money by having more programs per diskette.
PERCOM'S DOUBLER II **$159.00**

NEW MODEM GAMES

Play Checkers, Chess, or Othello over your modem with your friends! Each game features full graphics, sound, and a chat mode permitting you to send messages and sounds (to get the other persons attention). Games are for two players, either one or two computers. Complete documentation. All three:

Tape .... **$14.95** **Disk** **$19.95**

SUPERMETER



The biggest problem in loading tapes has been the volume control. Prerecorded tapes are produced at differing volume levels. Now finally, a device to let you set the correct volume levels for loading any tape. You will now load any tape the FIRST time. **SUPERMETER** plugs in (no cutting or soldering) between your tape recorder and the computer and lets you set the volume to the level that your computer wants.

SUPERMETER **$29.00**

Computer Shack

433

1691 Eason ● Pontiac, Michigan

New Games ● Utilities ● Business Programs WANTED

Dealer Inquires Invited.

C.O.D. - certified check M.O. or cash only, Sorry, no C.O.D. over $150.00! Most orders shipped next day. All orders must have shipping included. Please add 2% or $2.50, whichever is higher, for shipping (unless otherwise stated). Out-of-Country orders, please add $10.00 additional shipping and handling, balance returned upon shipment.

PHONE ORDERS WELCOME: (313) 673-2224 673-8700
Call between 8-2 E.S.T.

See List of Advertisers on page 386

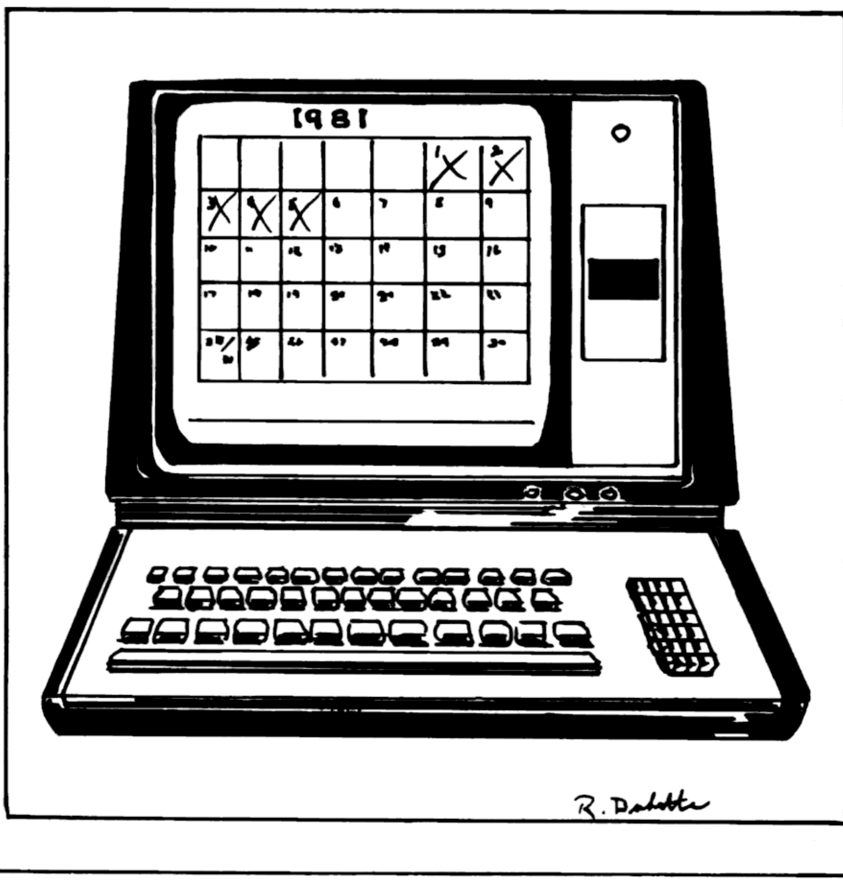
80 CALENDAR

October

Oct. 2 and 3 A two-day conference on **"Classroom Applications of Computers"** will be conducted at Independence High School, San Jose, CA, by Computer-Using Educators and Santa Clara Valley Mathematics Assn. The Conference will cover mathematics, science, business, music, special education, language arts and administrative applications of computers at levels ranging from pre-school through college. Information is available from Computer-Using Educators, Independence High School, 1776 Educational Park Dr., San Jose, CA 95133.

Oct. 24 and 25 The second annual **New Jersey Microcomputer Show and Fleamarket** is scheduled at the Holiday Inn North Convention Center, Newark, NJ. The show will feature 75 commercial exhibitors and more than 100 outdoor fleamarket vendors and user-group meetings for TRS-80s. Admission for the show and fleamarket is $5; for the fleamarket only, $3. It is sponsored by Kengore Corp., 3001 Rte. 27, Franklin NJ 08823.

Oct. 26–Nov. 4 Virginia Polytechnic Institute and State University will conduct three workshops in October and November. **Digital Electronics for Automation and Instrumentation** will be Oct. 26–28; **Microcomputer Design Interfacing and Programming using Z80/8085/8080** will be Oct. 29–31; and **Scientific Instrument Automation, Interfacing and Programming using the TRS-80 microcomputer** will be Nov. 2-4. Information is available from Dr. Lindy Leffel, CEC, Virginia Tech, Blacksburg, VA 24061.



Technical Education Research Centers, 8 Eliot St., Cambridge, MA 02138, will sponsor a series of workshops on **Microcomputers in Education"** in College Park, MD, **Oct 2-4;** Toronto, Canada, **October 22-24;** and Cambridge, MA, **Nov. 22-24.** They will include a discussion of educational issues, three microcomputer languages and laboratory and science applications of microcomputers.

The commission on Software Issues in the '80s, 1301 20th St., N.W., Suite 116, Washington, DC 20036, will sponsor the **National Conference on Software: Critical Decisions,** at the Shoreham Hotel, Washington, DC, **Oct. 5-6.** Subject areas will include education, software protection, technical issues and taxation.

The University of California, Berkeley, will sponsor a two-day course on **"Bit-Slice Microprocessor Design"** for engineers and engineering managers at the Marriott Santa Clara Hotel, Santa Clara, CA **Oct 26-27.** Fee is $350. Interested parties can register through the Office of Continuing Education in Engineering, University of California Extension, 2223 Fulton St., Berkeley, CA 94720.

Ken Orr and Associates, Inc., 715 E. 8th St., Topeka, KS 66607, will sponsor courses and conferences on several issues in October. Subjects are: **Oct.**

5-7, Data Structured Systems Design Users Conference, Topeka; Oct. 8, Systems Maintenance Update Conference, Topeka; Oct. 19-23, Structured Systems Design, Kansas City, MO; Oct. 20-23, Structured Requirements Definition, Atlanta, GA; Oct. 27-29, Structured Data Base Design, Topeka; Oct. 27-30. Structured Requirements Definition, Denver, CO.

November

The University of California, Berkeley, will sponsor a two-day course titled **"Comparison of Recent Microcomputer Architectures"** for design engineers, programmers and technical managers at the San Francisco Airport Hilton, **Nov. 9-10.** Fee is $400. Information and registration is available through the office of Continuing Education in Engineering, University of California Extension, 2223 Fulton St., Berkely, CA 94720

The New Mexico Computer Society will host the third annual **New Mexico Computer Fair Nov. 14** at the Albuquerque Civic Auditorium. Admission is free. Information is available from Ron Benninghoff, c/o New Mexico Computer Society, 515 Wyoming NE No. 2, Albuquerque, NM 87108

Ken Orr and Associates, Inc., 715 E. 8th St., Topeka, KS 66607, will sponsor several courses and conferences in November. Subjects are: **Nov. 3-6 Structured Requirements Definition, San Antonio, TX; Nov. 3-6, Structured Program Design, Boston, MA; Nov. 10, Management Overview of Data Structured Systems Development, Denver, CO; Nov. 11, Management Overview of Data Structured Systems Development, Portland, OR; Nov. 13, Management Overview of Data Structured Systems Development, Seattle, WA; Nov. 9-13, Structured Systems Design/Structured Program Design Combined Course, St. Louis, MO; Nov. 10-13, Structured Requirements Definition, Los Angeles, CA; Nov. 16-20, Structured Systems Design/Structured Program Design Combined Course, Washington, DC.**

December

Ken Orr and Associates, Inc., 715 E. 8th St., Topeka, KS 66607 will run several courses in December. Subjects are: **Dec. 1-4, Structured Systems Design, Chicago, IL; Dec. 8-10, Structured Program Design for Teleprocessing, Topeka, KS; Dec. 8-11, Structured Requirements Definition, Philadelphia, PA; Dec. 8-11, Structured Program Design, Kansas City, MO; Dec. 14-18, Structured Systems Design/Structured Program Design Combined Course, Atlanta, GA.**

See List of Advertisers on page 386

Leave typing to typists. Enter data the easy way.

Copyit

Ron Balewski
412 E. Ridge St.
Nanticoke, PA 18634

Program Listing 1. Copyit

```
                ; ***** COPYIT *****
                ; BY RON BALEWSKI
                ; 12/19/1980
4020            CSRPOS  EQU     4020H
0000            KEYSCN  EQU     0
0001            BREAK   EQU     1
                ; SHDOWN CAN BE CHANGED TO ANY DESIRED TURN-ON CODE
001A            SHDOWN  EQU     26
40A4            BASBEG  EQU     40A4H
4020            CSRPOS  EQU     4020H
0000            KEYSCN  EQU     0
0001            BREAK   EQU     1
001A            SHDOWN  EQU     26
40A4            BASBEG  EQU     40A4H
005F            CURSOR  EQU     95              ;CURSOR CHARACTER
42E9                    ORG     42E9H
                ;
                ;
                ;
                ;ENTRY POINT-- IF COPY IN PROGRESS, GO TO CONSEN
                ;              CALL RADIO SHACK KEYBOARD DRIVER
                ;              IF CHARACTER RETURNED = SHIFT-DOWN-ARROW
                ;                 THEN GO TO START A COPY OTHERWISE GIVE
                ;                 THE CHARACTER TO BASIC
                ;
42E9 3A6443     COPYIT  LD      A,(DUPFLG)      ;COPY IN PROGRESS?
42EC B7                 OR      A
42ED 2060               JR      NZ,CONSEN       ;IF SO, CONTINUE SENDING
42EF CD0000     KEYADD  CALL    KEYSCN          ;CALL R S KEYBOARD DRIVER
                ; THE ADDRESS OF THE ABOVE CALL WILL BE LOADED IN DURING
                ; THE PROGRAM SETUP TIME.
42F2 FE1A               CP      SHDOWN          ;SHIFT-DOWN PRESSED?
42F4 C0                 RET     NZ              ;IF NOT, RETURN
                ;
                ;
                ;
                ; START COPY AFTER SHIFT-DOWN PRESSED
                ;
                ; ASK FOR AND DECODE DESIRED LINE NUMBER.
                ; IF 0, LEAVE COPYIT
                ; IF BREAK PRESSED DURING INPUT, LEAVE COPYIT WITH A
                ;      BREAK CHARACTER
42F5 D9                 EXX                     ;SWAP REGS
42F6 216F43             LD      HL,MSG1         ;POINT TO COPY LINE MSG
42F9 CDA728             CALL    28A7H           ;PRINT IT
42FC 216543             LD      HL,COPBUF       ;GET ADD OF MY INPUT BUFF
42FF 0608               LD      B,8             ;MAX OF 8 CHARACTERS
4301 CDD905             CALL    05D9H           ;CALL INPUT RTN
4304 FE01               CP      BREAK           ;QUIT W/ BREAK?
4306 2845               JR      Z,EXIT          ;IF SO, LEAVE W/ BREAK
4308 3E5F               LD      A,CURSOR        ;GET CURSOR
430A 322240             LD      (4022H),A       ;TURN CURSOR BACK ON
430D CD5A1E             CALL    1E5AH           ;CONVERT # TO BINARY
4310 7A                 LD      A,D             ;0 RETURNED?
4311 B3                 OR      E
4312 2839               JR      Z,EXIT          ;IF SO, LEAVE
                ;
                ;
                ;  BEGIN SEARCHING BASIC TEXT FOR DESIRED LINE NUMBER
                ;  IF PRESENT, GO TO FOUND ELSE GO TO NOTFND
                ;
```

Program continues

Even Christmas can be tedious. About a week before the last Yule, I typed some seasonal music on my TRS-80. Because most songs contain more than one verse, and Christmas tunes are no exception, I found myself typing identical data statements over and over—not something I enjoy.

Since I had already typed the data once, I figured all I had to do was find it.

Copyit

I wrote a short utility (2K plus) that can save you time retyping long, tedious data statements when they are repeated in any Basic program. I call it Copyit.

Copyit lets you duplicate one line of Basic code in another. I placed it in low memory (below Basic) to eliminate the need for separate programs for 16, 32 and 48K machines. I patched Copyit into the keyboard's device control block (DCB), so that each time you strike a character, Basic calls Copyit, instead of the usual input routine.

Copyit checks a flag that indicates you need a copy of a previous data statement. If the flag says no, Copyit calls the key-

THE BOOKKEEPERS

FOR INFO CALL (603)-447-2745

Full Charge Bookkeeper—48K, 3 DRIVE, w/ALPHA **$129.95**

Intermediate Bookkeeper—48K, 2DRIVE & Printer **$109.95**

Cheap Bookkeeper—32K, 2DRIVE & Printer **$ 89.95**

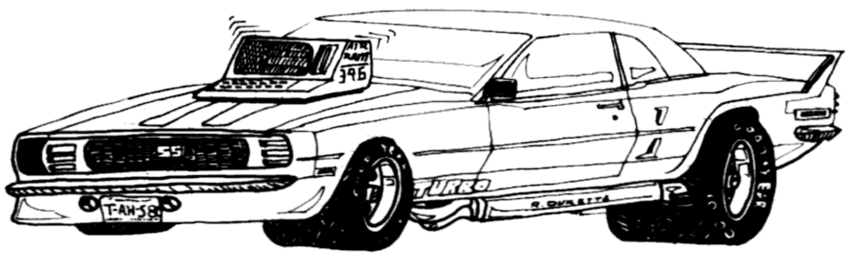
All Above Are Daily Journal—G/L Systems

Ann Rose, our Accounts Receivable Clerk—
48K, 2 DRIVE & PRINTER ______________ $150.00

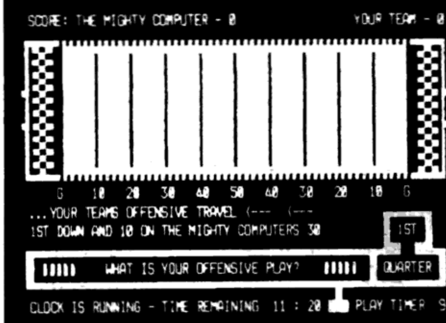
STURDIVANT & DUNN, INC. ✓82
BOX 277, 124 WASHINGTON ST., CONWAY, NH, 03818

101 USES FOR A DEAD MICROCOMPUTER

no.92



SUPER MICRO-PRO-CESSOR FOOTBALL



Animated Graphics — Game Time Clock (48K-32K) Available for TRS80* LII M1 or M3 48K DISC $22.95 — 32K DISC $19.95 (48K has more of everything!!!)

16K DISC OR TAPE $12.95

Many more super programs. Please write for free listing and program details

Attention: Software Authors—We want good programs—Top royalty plan. Dealer inquiries welcomed

* TRS80 is a Reg. Trademark of Tandy Corp

MICRO PRO SYSTEMS
Route #2 Box 533
Cumming, Georgia 30130
✓495

MAKE $$$$$ $$$$ MONEY

Selling 80 Microcomputing, the only major journal for the users of the TRS-80*, is a sure bet for getting the computer enthusiast into your store. Once through the door you can sell him anything.

We know "80" will make you money . . . it's the only magazine for the TRS-80* users and you know how many of those there are. So call today and join the dealers who make money with "80".

For information on selling 80 Microcomputing, call 603-924-7296 and speak with Ginnie Boudrieau, our Bulk Sales Manager. Or write to her at 80 Microcomputing, Pine Street, Peterborough, NH 03458.

***TRS-80 is a trademark of the Tandy Corp.**

✓See List of Advertisers on page 386

board driver.

If you pressed any character other than a shift down-arrow Basic records the character in the A register as usual.

If the keyboard driver returns the shift down-arrow, you'll activate Copyit and it will ask you for a line number. The routine converts this number to binary and searches for it. If Copyit can't find the line, it prints a message to that effect and terminates itself, until you activate it again.

If the routine finds the line number, it sets a "next-byte-to-copy" pointer to the first character of that line, turns on the copy flag, sets the A register to zero and returns control to Basic. Setting the A register to zero fools the computer into thinking that nothing has happened.

From this point on, whenever the keyboard is scanned and the copy flag is on, Copyit reads the "next-byte-to-copy" pointer. It loads this byte into the A register, and the pointer is bumped up one. Control is then returned to Basic with the character in the A register, just as if you had typed it.

Since the copy flag remains on, the entire line will be sent to Basic, character by character, until the "next-byte-to-copy" is zero. If so, Copyit has found the end of the line; it turns off the flag, and Basic resumes.

Program Listing 1 Continued

```
4314 DD2AA440  00470          LD      IX,(BASBEG)       ;POINT TO BEGIN OF BASIC
4318 DD6E02    00480 PROLIN   LD      L,(IX+2)          ;GET CURRENT LINE #
431B DD6603    00490          LD      H,(IX+3)
431E DF        00500          RST     18H               ;COMPARE WITH # WANTED
431F 280D      00510          JR      Z,FOUND           ;JUMP IF FOUND
4321 3020      00520          JR      NC,NOTFND         ;IF CURRENT>WANTED
               00530                                    ;   THEN NOT IN TEXT.
4323 DD6601    00540          LD      H,(IX+1)          ;GET ADD OF NEXT LINE
4326 DD6E00    00550          LD      L,(IX+0)
4329 E5        00560          PUSH    HL                ;TRANSFER TO
432A DDE1      00570          POP     IX                ;    POINTER
432C 18EA      00580          JR      PROLIN            ;GO PROCESS THIS LINE
               00581 ;
               00582 ;
               00583 ;
               00584 ;FOUND-   SET DUPLICATION FLAG (DUPFLG) AND SET POINTER FOR
               00585 ;         THE NEXT CALLS TO FETCH THE TEXT THEN LEAVE.
432E 3E01      00590 FOUND    LD      A,1               ;SET COPY IN PROGRESS FLG
4330 326443    00600          LD      (DUPFLG),A
4333 DD23      00610          INC     IX                ;BUMP PAST
4335 DD23      00620          INC     IX                ;    NEXT LINE PTR
4337 DD23      00630          INC     IX                ;BUMP PAST
4339 DD23      00640          INC     IX                ;    LINE #
433B DD226243  00650          LD      (CURBYT),IX       ;POINT TO WHERE TO START
               00660                                    ;    COPYING FROM
433F 3E00      00670          LD      A,0               ;SAY NOTHING HAPPENED
4341 180A      00680          JR      EXIT              ;GO BACK TO BASICS
               00681 ;
               00682 ;
               00683 ;
               00684 ;NOTFND-  PRINT NOT FOUND MESSAGE AND LEAVE
4343 217C43    00690 NOTFND   LD      HL,MSG2           ;POINT TO NOT FOUND MSG
4346 CDA728    00700          CALL    28A7H             ;PRINT IT
4349 3E00      00710          LD      A,0
434B 1800      00720          JR      EXIT              ;LEAVE
434D D9        00840 EXIT     EXX                       ;RE-SWAP REGS
434E C9        00850          RET
               00860 ;
               00870 ;
               00880 ; CONSEN-  CONTINUE SENDING THE CURRENT LINE BEING COPIED
               00890 ;             BYTE BY BYTE 'TILL IT'S DONE
               00900 ;
434F D9        00910 CONSEN   EXX                       ;SWAP REGS
4350 2A6243    00920          LD      HL,(CURBYT)       ;POINT TO BYTE TO BE SENT
4353 7E        00930          LD      A,(HL)
4354 B7        00940          OR      A                 ;BYTE TO BE SENT 0?
4355 2806      00950          JR      Z,QUIT            ;IF SO, LINE ENDED, QUIT
4357 23        00960          INC     HL                ;BUMP TO NEXT CHR
4358 226243    00970          LD      (CURBYT),HL       ;    & PUT FOR NEXT TIME
435B D9        00980          EXX                       ;RESWAP REGS
435C C9        00990          RET                       ;BACK TO BASIC
435D 326443    01000 QUIT     LD      (DUPFLG),A        ;TURN OFF COPY FLAG
4360 D9        01010          EXX                       ;RE-SWAP REGS
4361 C9        01020          RET                       ;BACK TO BASIC
4362 0000      01030 CURBYT   DEFW    0                 ;NEXT-TO-SEND POINTER
4364 00        01040 DUPFLG   DEFB    0                 ;DUPLICATION FLAG
000A           01050 COPBUF   DEFS    10
436F 20        01060 MSG1     DEFM    ' COPY LINE? '
437B 00        01070          DEFB    0
437C 20        01080 MSG2     DEFM    ' LINE NOT FOUND '
438C 00        01090          DEFB    0
               01091 ;
               01092 ;
               01093 ;
               01094 ;SETUP-   THIS SECTION IS OVERWRITTEN BY YOUR BASIC.
               01095 ;         IT SETS START-BASIC POINTER TO ITS BEGINNING,
               01096 ;         PATCHES COPYIT INTO THE KEYBOARD DCB, AND
               01097 ;         RETURNS TO BASIC WITH A "NEW" COMMAND.
438D 218D43    01100 SETUP    LD      HL,SETUP          ;WHERE BASIC WILL START
4390 3600      01110          LD      (HL),0            ;INITIALIZE TO 0
4392 23        01120          INC     HL
4393 22A440    01130          LD      (40A4H),HL        ;PUT IN BASIC PGM PTR
4396 21191A    01140          LD      HL,1A19H          ;GET READY ADD
4399 E5        01150          PUSH    HL                ;PUT READY ON STACK
439A 2A1640    01160          LD      HL,(4016H)        ;GET KBD DRIVER ADD
439D 22F042    01170          LD      (KEYADD+1),HL     ;SAVE IT FOR USE
43A0 21E942    01180          LD      HL,COPYIT         ;GET MY ADDRESS
43A3 221640    01190          LD      (4016H),HL        ;MAKE ME NEW DRIVER
43A6 C34D1B    01200          JP      1B4DH             ;GO NEW AND READY
438D           01210          END     SETUP
```

Operating Instructions

Let's take an example. To copy line 10 into line 20, type: 20 shift down-arrow. Copyit will respond with: Line Number?

You then type 10 and hit Enter. Hit Enter once again, and you have rejoined Basic.

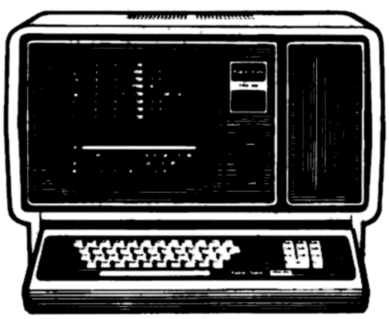
If your machine has a new ROM, you will have to type: shift down-arrow Z to activate Copyit. You can tell if you have a new ROM, because your computer will ask you for the Memory Size while, if your computer has an old ROM, it will ask you for the Memory Size. You can change this by replacing the 26 in line 130 with the ASCII code you want to use.

You can activate Copyit at any time, not just at the beginning of a line. You can type a statement or two before copying line 10; you can also copy something after copying line 10, but before entering the line; or you can copy line 10 into line 20 more than one time.

Once you see how much typing Copyit can save you, you'll start loading it habitually before typing long data statements. Remember, you may have to move those statements. ■

THE BIGGEST NAME IN LITTLE COMPUTERS™

TRS-80™ Model II—Your Best Buy In a Business Microcomputer



UP TO 15% OFF! on

TRS-80™ computers, software and peripherals

Similar values on all merchandise

CALL COLLECT:

915-283-2920

Van Horn Office Supply ✓137

701 W. Broadway – P O Box 1060

Van Horn, Texas 79855

DEALER GO55

Form F48 Provided

Standard Warranty in Effect

THE NATIONWIDE SUPERMARKET OF SOUND®

ELECTRIC SPREADSHEET™

NEW - For Personal or Business Use

ELECTRIC SPREADSHEET does everything you can do with a pencil, paper and calculator. Start with a blank screen. Move the fast cursor to desired location. Enter line or column labels or data. Select preprogrammed operators for line, column, or cell calculations. Run. Results appear on the screen, formatted for your printer. Revise. Run again.

PREPROGRAMMED OPERATORS for mathematics, finance, and statistics. Save data on tape or disk. The screen is your window to a larger spreadsheet. Output features scrolling and split-screen. Variable formats for labels and data.

UNLIMITED APPLICATIONS. P/L forecast. Personal budget. Real estate investment. Net worth forecast. Invoice. Cash flow estimate. Sales analysis. Check record. Business forms.

16K version has 50 operators. 32/48K version has 70 operators plus histogram plot, remote storage of data, global revise spreadsheet layout, alpha entries, and more.

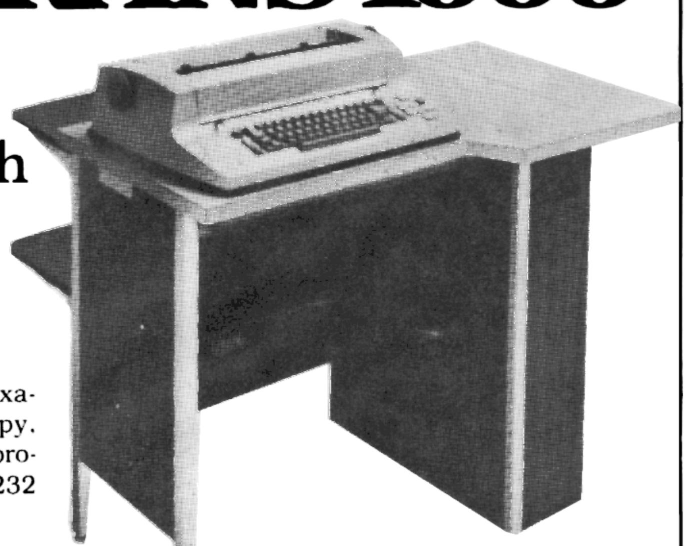
16K Model I or III Tape	$34.95
32/48K Model I or III Tape	$64.95
48K Model I or III Disk (Specify Model)	$67.95

YOU CAN'T LOSE! Calif. orders add 6½% tax.

Dan G. Haney & Associates, Inc. ✓464
P.O. Box 687, San Mateo, CA 94401
(415) 493-4094

The DATA-TRANS 1000

A completely refurbished IBM Selectric Terminal with built-in ASCII Interface.



***FOR YOUR TRS-80 WITH OR WITHOUT EXPANSION INTERFACE. AVAILABLE WITH CENTRONICS TYPE PARALLEL PORT.**

Features:

- 300 Baud Serial
- 14.9 characters per second printout
- Reliable heavy duty Selectric mechanism
- RS-232C Interface
- Documentation included
- 60 day warranty—parts and labor
- High quality Selectric printing Off-line use as typewriter
- Optional tractor feed available
- 15 inch carriage width

Also works with Exatron's Stringy floppy, for fast loading of programs. (Has RS232 built in stringy)

HOW TO ORDER DATA-TRANS 1000

1. We accept Visa, Master Charge. Make cashiers checks or personal check payable to:

DATA-TRANS

2. All orders are shipped F.O.B. San Jose, CA
3. Deliveries are immediate

Desk and table top models also available.

For orders and information

DATA-TRANS

45277 Fremont Blvd., #7 ✓274
Fremont CA 94538

Phone: **(408) 263-9246**

Take pride in your listing's appearance.

Listprog

Doug Riffel
1836 Metzerott Road #906
Adelphi, MD 20783

I have always been a stickler for neat and legible program documentation. Consequently, I quickly became frustrated at the total lack of form control incorporated in the Basic LLIST command. Even NEWDOS/80 with all its extensions to Disk Basic offers no help.

There were two basic features I wanted. I quickly found that automatically feeding the page over the perforation was not as easy as I thought it would be. I also wanted to indent the lines to provide room for three-hole punching so the program listings could be stored in binders.

I also wanted some other neat little goodies. First, the name of the program should appear prominently in the header. Second, I wanted to print the date the program is listed. Third, I needed page numbering and finally, double-spacing between statements to provide for pencil corrections.

All these features have been incorporated in Program Listing 1, which I call Listprog. It requires Disk Basic and NEWDOS/80 to operate, and will work equally well with TRSDOS with the deletion of line 1000.

Although Listprog was written to capitalize on some of the features of Radio Shack's Line Printer IV (Centronics 737), with some minor modifications it will work just as well with other printers. The necessary modifications will be covered later.

```
10 CLS:CLEAR 500:PN=1
20 POKE 16425,1:PT=100:IN=15
1000 IF LEFT$(TIME$,2)="00" THEN INPUT"Enter todays date  (MM/DD
/YY)";A$:IF LEN(A$)=8 THEN A$="DATE "+A$:CMD"A$"
1010 CLS:INPUT"FILE NAME";FL$
1020 OPEN "I",1,FL$
1030 SL=INSTR(FL$,"/")-1:IF SL<1 THEN SL=LEN(FL$)
1040 LPRINTCHR$(27);CHR$(19);CHR$(27);CHR$(14);TAB(19);LEFT$(FL$
,SL);CHR$(27);CHR$(15);TAB(42);"DATE: ";LEFT$(TIME$,8):LPRINTCHR
$(138)
1050 LPRINTCHR$(138):LPRINTCHR$(27);CHR$(20);TAB(60);"* PAGE ";P
N;" *":LPRINTCHR$(138):LPRINTCHR$(138)
1060 IF EOF(1) LPRINTCHR$(27);CHR$(19);CHR$(11):CLOSE:PRINT:PRIN
T"* PROGRAM TERMINATED *":END
1070 LINE INPUT#1,A$
1080 SP=INSTR(A$," "):TS=SP+1:TL=LEN(A$)-SP:NL=INT(TL/PT):EC=TL-
NL*PT
1090 FOR N=TS TO LEN(A$)
1100 IF MID$(A$,N,1)=CHR$(10) THEN MID$(A$,N,1)=CHR$(94)
1110 NEXT
1120 LPRINTTAB(IN);LEFT$(A$,SP);
1130 IF NL=0 AND EC=0 THEN LPRINTCHR$(138):GOTO 1170
1140 IF NL=0 LPRINTTAB(IN+8);MID$(A$,TS,EC):LPRINTCHR$(138):GOTO
 1170
1150 LPRINTTAB(IN+8);MID$(A$,TS,PT)
1160 TS=TS+PT:NL=NL-1:GOTO 1130
1170 IF PEEK(16425)>55 THEN LPRINTCHR$(11):PN=PN+1:GOTO 1050
1180 GOTO 1060
```

Program Listing

Listprog

Whenever a printed line exceeds the line length of the printer, the printer logic automatically prints across a full line and then prints the remainder on the following line. When this happens, location 16425 (Basic's line counter) does not get updated. This, in essence, means total loss of form control.

A similar situation occurs if the down arrow is encountered within a statement. I use the down arrow quite frequently to print multiple lines on the screen with one statement. If the printer logic detects this character, the line up to that point is printed and the remainder is printed on the following line. Again, the Basic line counter is not updated.

When a program is stored on disk in ASCII format, the first space encountered in a statement is always the separator between the statement number and the statement text.

Lines 10 and 20 of Listprog merely clear string space and set certain variables. Line 1000 (NEWDOS/80 only) examines TIME$ to see if a date has been entered. If not, it requests the date and executes a DOS command to set it.

Lines 1010 and 1020 request the filespec and open that file.

Line 1030 examines the filespec for the presence of a slash. If a slash is present, it and everything to the right of it (the file extension) is discarded. I personally prefer to keep ASCII-formatted program files separate from the Basic program files by using the filespec extension "/TXT"; therefore, the extension on the filespec actually being printed is irrelevent.

Line 1040 prints the header which consists of the program name, printed at five characters per inch, and the date, printed at 10 characters per inch. For any printer other than the LP IV, this line should be modified accordingly.

Line 1050 turns on the 16.7 characters per inch mode (132

✓See List of Advertisers on page 386

characters per line for the LP IV) and prints the page number. This line should also be modified for printers other than the LP IV.

Line 1060 returns the printer to 10 characte per inch, ejects the last je and terminates Listpro. .ine 1070 is the required form ror inputting ASCII-formatted program files.

Line 1080 provides the necessary set-up information for the remainder of the program. There are really two left margins, one for the statement number and one for the statement text. The length of the printed statement text line is predetermined by the value assigned to variable PT in line 20. The left margin for the statement number is also predetermined in line 20 by variable IN. For printers with line lengths different than the LP IV, these variables may be changed; however, IN + 8 + PT cannot exceed the line length of the printer.

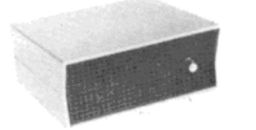
Other variables and their use are as follows:

- SP = Location of the space separator between line number and text
- TS = Beginning of statement text
- TL = Length of statement text
- NL = Number of full PT-character lines to be printed
- EC = Number of characters in last line if less than PT

Lines 1090–1110 search the statement text for the down arrow (CHR$(10)) and replace it with the ASCII carat (CHR$ (94)). If the printer you are using cannot print this character, you will want to change these lines.

Lines 1130–1170 perform the actual printing, feeding properly over the paper perforations.

Listprog has become an extremely popular program on my TRS-80 and I use it to document all my Basic programs. The program logic is quite simple and lends itself to easy tailoring for individual preferences. ■



SPEAK-EAZY

the :HUMAN QUALITY: "VOICE"

for your TRS-80* that is so good you won't believe it

Phone for demo. $299.00

6 programs on tape or disk. + user manual

-NEW-

DOUBLE DENSITY DISK CONTROLER WITH 16-K ONBOARD RAM FOR YOUR MODEL I. NO EXPANSION INTERFACE NEEDED, RUNS SINGLE OR DOUBLE DEN.

$299.00

-NEW-

$29.00 SUPER SENSITIVE LIGHT PEN

SIMPLIFY YOUR TAPE LOADING WITH :cload: THE LED ·VU· METER

$29.00

TRS-80 IS A TRADEMARK OF RADIO SHACK

PROGRESSIVE ELECTRONICS ✓277

643 E. CHESTNUT ST.

LANCASTER OH. 43130

PHONE - 614-687-1019

LOW PRICES ON TRS-80*

From Radio Ranch, Inc.

In business 27 years at the same location providing electronic equipment and service.

We Are Better Equipped To Provide

TRS-80* COMPUTERS & Accessories Software

TRS-80* Mod II - 64K	3440.00
TRS-80* Mod III - 32K, 2 Dr	2195.00
TRS-80* Line Printer 6	1020.00
CORVUS** 5-10-20 M Hard Disks	Call For Prices

"CORDOS" and "OASIS" operating systems

Full Factory Warranty · Complete Service Department

Radio Ranch, Inc. ✓133

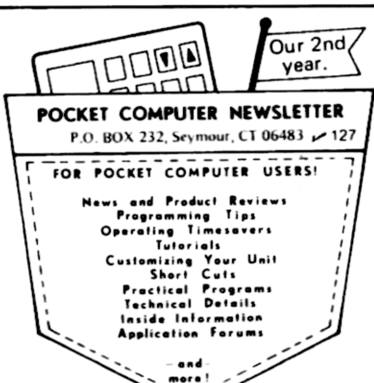
Cert. Check
Cashiers Ck.

Route 3, Radio Ranch Airport
Polo, Illinois 61064

Call Toll Free 800-435-8310

In Ill. Call 815-946-2371

*Trademark Tandy Corp. **Trademark Corvus Systems, Inc.



FREE PREMIUM! 1982 subscribers will receive the special MATH1 package of PC programs. You qualify to receive this valuable free gift if you check either box 2 or 3 below! Don't delay! Get started learning how to enjoy your pocket computer today. Use this handy subscription card!

SUBSCRIPTION FORM

- ☐ 1981 Charter Subscriber (Issues 1 – 10). $20.00 for U.S. delivery. (U.S. $24.00 to Canada. U.S. $30.00 elsewhere.)
- ☐ 1981/82 Charter Subscriber (Issues 1 – 20). $40.00 in U.S. (U.S. $48.00 to Canada. U.S. $60.00 elsewhere.)
- ☐ 1982 Regular Subscriber (Issues 11 – 20). $30.00 in U.S. (U.S. $36.00 to Canada. U.S. $45.00 elsewhere.)
- ☐ Sample issue. $3.00 in U.S. (U.S. $4.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: ______________________________

Addr: ______________________________

City: ________________ State: ___ Zip: ______

MC/VISA # _______________ Expires: ______

Signature: ______________________________

MODELS I AND III

ALL PROGRAMS USE TESTED

MICRO-TYPIT A text editing program that uses the keyboard "as is" like a typewriter. All edit and prompt functions are "built in" so instruction or learning is minimum. Text can be generated three pages at a time and printed either numbered, or unnumbered single or double spaced. Also, right justification is optional. Does not require substitution of @ for commas or any other character revision. Slowest function is transferring text to and from tape. Neat error free text with large print titles. Excellent for specifications, agreements, instructions, form letters or announcements. Complete with sound for TRS-80 Model I or III and Microteck printer easily adaptable to most other printers. PRICE.....$25.00

CLIENT FILES Allows keeping several contracts for up to 20 clients. Enter contracts including either payment amounts or percentages. As bills are sent out and payments received, the program is a record by contract of amounts outstanding and not billed including a summary of receivables and work in progress. Works well with one to 3 projects for a dozen or so clients. PRICE.....$25.00

ENGINEERING PROGRAMS

LIGHTING CALCULATIONS Calculates the required number of fixtures for a room given the dimensions and reflectances by the zone cavity method. Three standard fixtures are "built in" so that 8 room calculations are made to compare results such as watts, footcandles and number of fixtures. PRICE.....$30.00

HVAC PROGRAM This program calculates sensible and latent cooling and heating load by zone each two hours of the typical hottest and coldest days. Equations per 1977 ASHRAE Fundamental Guide page 25.2. Three programs are: Calculations, Geographic and Variable Data and Data Generator. PRICE.....$35.00

All tapes furnished with full documentation and instructions. Documentation, Instructions and Source List alone.....$15.00

All programs on tape require Level II and 16K. Printer is required for MICRO-TYPIT and optional on all others. Do not forget to include the Model Number (I or III) with your order.

Send Check or Money Order (no COD's please) to: ✓52

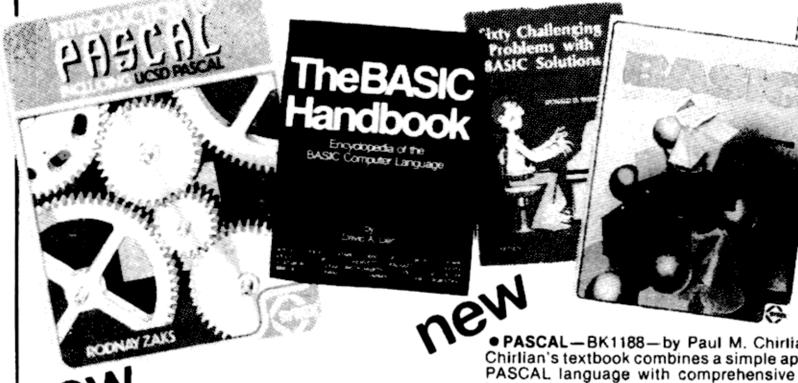
Coolidge & Associates

1317 S. CONGRESS #10 AUSTIN, TEXAS 78704 (512) 443-5329

✓See List of Advertisers on page 386

80 microcomputing™ bookshelf

—BASIC & PASCAL—



new

new

● **INTRODUCTION TO TRS-80 LEVEL II BASIC AND COMPUTER PROGRAMMING**—BK1219—by Michael P. Zabinski. Written by an experienced educator, this is the book for those beginners who want to learn about computers without having to become an expert. It has practical programs, useful line-by-line comments, excellent flowcharts accompanied by line numbers and over 200 exercises which help the reader assess progress, reinforce comprehension, and provide valuable practical experience. $10.95.*

● **50 BASIC EXERCISES**—BK1192—by J. P. Lamoitier. This book is structured around the idea that the best way to learn a language is through actual practice. It contains 50 completely explained exercises: statement and analysis of the problem, flowcharts, programs and actual runs. Program subjects include mathematics, business, games, and operations research, and are presented in varying levels of difficulty. This format enables anyone to learn BASIC rapidly, checking their progress at each step. $12.95*

● **THE BASIC HANDBOOK**—BK1174—**NEW, SECOND EDITION** by David Lien. This book is unique. It is a virtual ENCYCLOPEDIA of BASIC. While not favoring one computer over another, it explains over 250 BASIC words, how to use them and alternate strategies. If a computer does not possess the capabilities of a needed or specified word, there are often ways to accomplish the same function by using another word or combination of words. That's where the HANDBOOK comes in. It helps you get the most from your computer, be it a "bottom-of-the-line" micro or an oversized monster. $19.95.*

● **LEARNING LEVEL II**—BK1175—by David Lien. Written especially for the TRS-80, this book concentrates on Level II BASIC, exploring every important BASIC language capability. Updates are included for those who have studied the Level I User's Manual. Sections include: how to use the Editor, dual cassette operation, printers and peripheral devices, and the conversion of Level I programs to Level II. $15.95.*

● **BASIC BASIC (2ND EDITION)**—BK1026—by James S. Coan. This is a textbook which incorporates the learning of computer programming using the BASIC language with the teaching of mathematics. Over 100 sample programs illustrate the techniques of the BASIC language and every section is followed by practical problems. This second edition covers character string handling and the use of data files. $10.50.*

● **ADVANCED BASIC**—BK1000—Applications, including strings and files, coordinate geometry, area, sequences and series, simulation, graphing and games. $10.75*

● **SIXTY CHALLENGING PROBLEMS WITH BASIC SOLUTIONS (2nd Edition)**—BK1073—by Donald Spencer, provides the serious student of BASIC programming with interesting problems and solutions. No knowledge of math above algebra required. Includes a number of game programs, as well as programs for financial interest, conversions and numeric manipulations. $6.95.*

● **PASCAL**—BK1188—by Paul M. Chirlian. Professor Chirlian's textbook combines a simple approach to the PASCAL language with comprehensive coverage on how a computer works, how to use a flowchart, working from a terminal as well as batch operation and debugging. Special attention is paid to idiosyncrasies of the language and syntax flowcharts abound for the convenience of the experienced programmer. Well indexed. $12.95*

● **INTRODUCTION TO PASCAL**—BK1189—by Rodnay Zaks. A step-by-step introduction for anyone wanting to learn the language quickly and completely. Each concept is explained simply and in a logical order. All features of the language are presented in a clear, easy-to-understand format with exercises to test the reader at the end of each chapter. It describes both standard PASCAL and UCSD PASCAL, the most widely used dialect for small computers. No computer or programming experience is necessary. $14.95.*

● **PROGRAMMING IN PASCAL**—BK1140—by Peter Grogono. The computer programming language PASCAL was the first language to embody in a coherent way the concepts of structured programming, which has been defined by Edsger Dijkstra and C.A.R. Hoare. As such, it is a landmark in the development of programming languages. PASCAL was developed by Niklaus Wirth in Zurich; it is derived from the language ALGOL 60 but is more powerful and easier to use. PASCAL is now widely accepted as a useful language that can be efficiently implemented, and as an excellent teaching tool. It does not assume knowledge of any other programming language; it is therefore suitable for an introductory course. $12.95.*

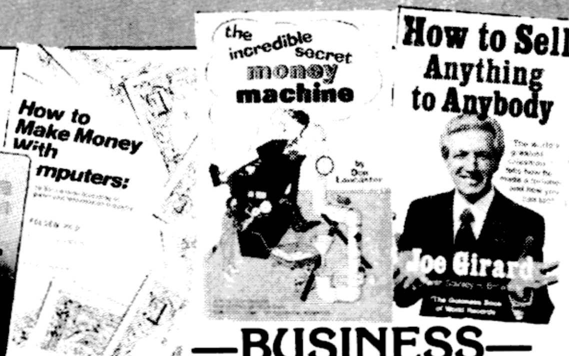
—GAMES—

● **40 COMPUTER GAMES**—BK7381—Forty games in all in nine different categories. Games for large and small systems, and even a section on calculator games. Many versions of BASIC used and a wide variety of systems represented. A must for the serious computer gamesman. $7.95*

● **BASIC COMPUTER GAMES**—BK1074—Okay, so once you get your computer and are running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. $7.50.*

● **MORE BASIC COMPUTER GAMES**—BK1182—edited by David H. Ahl. More fun in BASIC! 84 new games from the people who brought you *BASIC Computer Games*. Includes such favorites as Minotaur (battle the mythical beast) and Eliza (unload your troubles on the doctor at bargain rates). Complete with game description, listing and sample run. $7.50.*

● **WHAT TO DO AFTER YOU HIT RETURN**—BK1071—PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, many illustrations. Lunar Landing, Hammurabi, King, Civel 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $10.95.*



—BUSINESS—

● **THEORY Z**—BK1226—How American Business Can Meet the Japanese Challenge—by William Ouchi. Why are the Japanese catching up and surpassing American industrial productivity? What allows Japanese industrialists to offer guaranteed lifetime employment to their workforce? This book will help you understand the Theory Z managerial philosophy and its implications for the American corporate future. Examples are given of the American industrial giants already operating under Z-style management, and the impact of this style on the quality of their executives and workers is explored. A must for the alert businessman, large or small. $12.95.*

● **SO YOU ARE THINKING ABOUT A SMALL BUSINESS COMPUTER**—BK1222—by Richard G. Canning and Nancy C. Leeper. For a well-organized manual on the process of selecting the right computer system for your small business, this text can't be excelled. Designed to introduce the novice in data and word processing to the real benefits of computerization, the book is filled with money- and time-saving tips, photos of equipment, lists of suppliers, prices, explanations of computer terminology, and helpful references to additional sources of information. Everyone contemplating a first computer installation should have this book. $14.00.*

● **PAYROLL WITH COST ACCOUNTING—IN BAS[illegible]** BK1001—by L. Poole & M. Borchers, includes pro[illegible] listings with remarks, descriptions, discussions [illegible] principle behind each program, file layouts, and [illegible] plete user's manual with step-by-step instructions, flowcharts, and simple reports and CRT displays. Payroll and cost accounting features include separate payrolls for up to 10 companies, time-tested interactive data entry, easy correction of data entry errors, job costing (labor of distribution), check printing with full deduction and pay detail, and 16 different printed reports, including W-2 and 941 (in **CBASIC**). $20.00.*

● **SOME COMMON BASIC PROGRAMS**—BK1053—published by Adam Osborne & Associates, Inc. Perfect for non-technical computerists requiring ready-to-use programs. Business programs, plus miscellaneous programs. Invaluable for the user who is not an experienced programmer. All will operate in the standalone mode. $14.99 paperback.

● **PIMS: PERSONAL INFORMATION MANAGEMENT SYSTEM**—BK1009—Learn how to unleash the power of a personal computer for your own benefit in this ready-to-use data-base management program. $11.95.*

—MONEYMAKING—

● **HOW TO MAKE MONEY WITH COMPUTERS**—BK1003—In 10 information-packed chapters, Jerry Felsen describes more than 30 computer-related, money-making, high profit, low capital investment opportunities. $15.00.*

● **HOW TO SELL ANYTHING TO ANYBODY**—BK7306—According to *The Guinness Book of World Records*, the author, Joe Girard, is "the world's greatest salesman." This book reveals how he made a fortune—and how you can, too. $2.25.*

● **THE INCREDIBLE SECRET MONEY MACHINE**—BK1178—by Don Lancaster. A different kind of "cookbook" from Don Lancaster. Want to slash taxes? Get free vacations? Win at investments? Make money from something that you *like* to do? You'll find this book essential to give you the key insider details of what is really involved in starting up your own money machine. $5.95.*

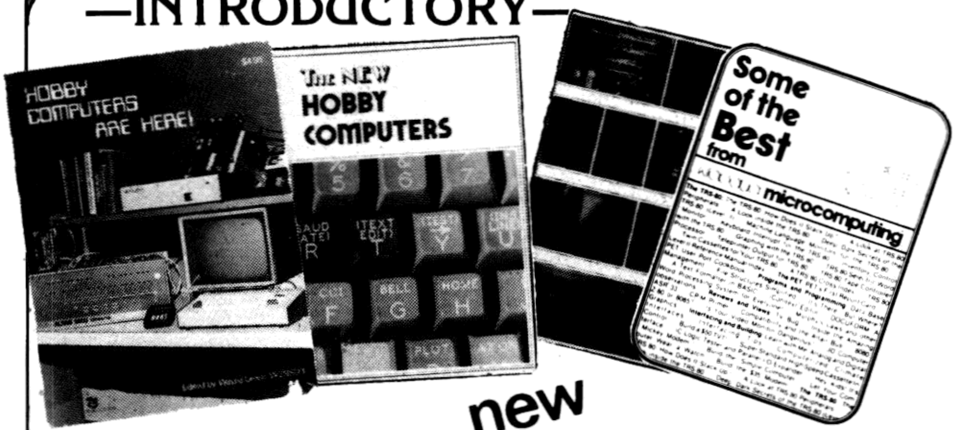
*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to 80 Microcomputing Bookshelf ● Peterborough NH 03458. Be sure to include check or detailed credit card information.

No C.O.D. orders accepted. All above add $1.50 handling. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

80 microcomputing™ bookshelf

—INTRODUCTORY—



new

● **UNDERSTANDING AND PROGRAMMING MICROCOMPUTERS**—BK7382—A valuable addition to your computing library. This two-part text includes the best articles that have appeared in 73 and Kilobaud Microcomputing magazines on the hardware and software aspects of microcomputing. Well-known authors and well-structured text helps the reader get involved. $10.95*

● **SOME OF THE BEST FROM KILOBAUD MICROCOMPUTING**—BK7311—A collection of the best articles that have appeared in Kilobaud MICROCOMPUTING. Included is material on the TRS-80 and PET systems, CP/M, the 8080/8085/Z-80 chips, the ASR-33 terminal. Data base management, word processing, text editors and file structures are covered too. Programming techniques and hard-core hardware construction projects for modems, high speed cassette interfaces and TVTs are also included in this large format, 200 plus page edition. $10.95*

● **YOUR FIRST COMPUTER**—BK1191—by Rodnay Zaks. Whether you are using a computer, thinking about using one or considering purchasing one, this book is indispensable. It explains what a computer system is, what it can do, how it works and how to select various components and peripheral units. It is written in everyday language and contains invaluable information for the novice and the experienced programmer. (The first edition of this book was published under the title "An Introduction to Personal and Business Computing".) $7.95*

● **MICROPROCESSOR INTERFACING TECHNIQUES**—BK1037—by Austin Lesea & Rodnay Zaks—will teach you how to interconnect a complete system and interface it to all the usual peripherals. It covers hardware and software skills and techniques, including the use and design of model buses such as the IEEE 488 or S-100. $15.95*

● **HOBBY COMPUTERS ARE HERE!**—BK7322—If you want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with fundamentals and explains the circuits, and the basics of programming, along with a couple of TVT construction projects, ASCII-Baudot, etc. This book has the highest recommendations as a teaching aid. $4.95*

● **THE NEW HOBBY COMPUTERS**—BK7340—This book takes it from where "HOBBY COMPUTERS ARE HERE!" leaves off, with chapters on Large Scale Integration, how to choose a microprocessor chip, an introduction to programming, low cost I/O for a computer, computer arithmetic, checking memory boards . . . and much, much more! Don't miss this tremendous value! Only $4.95*

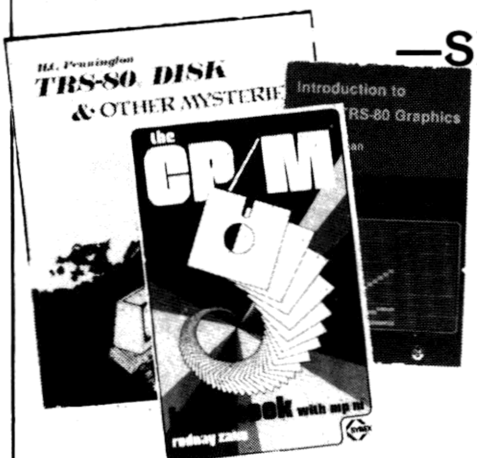
● **AN INTRODUCTION TO MICROCOMPUTERS, VOL. 0**—BK1130—The Beginner's Book—Written for readers who know nothing about computers—for those who have an interest in how to use computers—and for everyone else who must live with computers and should know a little about them. The first in a series of 4 volumes, this book will explain how computers work and what they can do. Computers have become an integral part of life and society. During any given day you are affected by computers, so start learning more about them with Volume 0. $7.95*

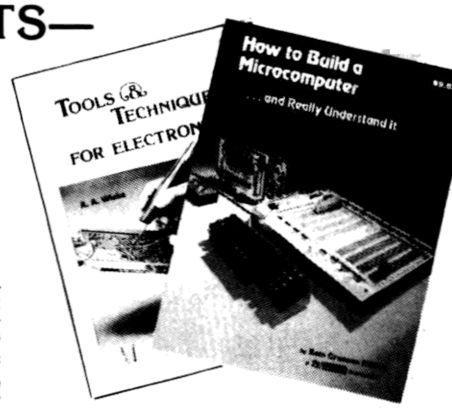
● **VOL. I**—BK1030—**2nd Edition completely revised.** Dedicated to the basic concepts of microcomputers and hardware theory. The purpose of Volume I is to give you a thorough understanding of what microcomputers are. From basic concepts (which are covered in detail), Volume I builds the necessary components of a microcomputer system. This book highlights the difference between minicomputers and microcomputers. $12.99*

● **VOL. II**—BK1040 (with binder)—Contains descriptions of individual microprocessors and support devices used only with the parent microprocessor. Volume II describes all available chips. $31.99*

● **VOL. III**—BK1133 (with binder)—Contains descriptions of all support devices that can be used with any microprocessor. $21.99*

—SPECIAL INTERESTS—





● **TRS-80 DISK AND OTHER MYSTERIES**—BK1181—by Harvard C. Pennington. This is the definitive work on the TRS-80 disk system. It is full of detailed "How to" information with examples, samples and in-depth explanations suitable for beginners and professionals alike. The recovery of one lost file is worth the price alone. $22.50*

● **MICROSOFT BASIC DECODED AND OTHER MYSTERIES**—BK1186—by James Farvour. From the company that brought you *TRS-80 DISK AND OTHER MYSTERIES*! Contains more than 6500 lines of comments for the disassembled Level II ROMs, six additional chapters describing every BASIC subroutine, with assembly language routines showing how to use them. Flowcharts for all major routines give the reader a real insight into how the interpreter works. $29.50.

● **THE CUSTOM TRS-80 AND OTHER MYSTERIES**—BK1218—by Dennis Kitsz. More than 300 pages of TRS-80 customizing information. With this book you'll be able to explore your computer like never before. Want to turn an 8 track into a mass storage unit? Individual reverse characters? Replace the BASIC ROMs? Make Music? High speed, reverse video, Level I and Level II? Fix it if it breaks down? All this and much, much more. Even if you have never used a soldering iron or read a circuit diagram, this book will teach you how! This is the definitive guide to customizing your 80! $29.95*

● **BASIC FASTER AND BETTER AND OTHER MYSTERIES**—BK1221—by Lewis Rosenfelder. You don't have to learn assembly language to make your programs run fast. With the dozens of programming tricks and techniques in this book you can sort at high speed, swap screens in the twinkling of an eye, write INKEY routines that people think are in assembly language and add your own commands to BASIC. Find out how to write elegant code that makes your BASIC really hum, and explore the power of USR calls. $29.95*

● **THE CP/M HANDBOOK (with MP/M)**—BK1187—by Rodnay Zaks. A complete guide and reference handbook for CP/M—the industry standard in operating systems. Step-by-step instruction for everything from turning on the system and inserting the diskette to correct user discipline and remedial action for problem situations. This also includes a complete discussion of all versions of CP/M up to and including 2.2, MP/M and CDOS. $14.95*.

● **INTRODUCTION TO TRS-80 GRAPHICS**—BK1180—by Don Inman. Dissatisfied with your Level I or Level II manual's coverage of graphics capabilities? This well-structured book (suitable for classroom use) is ideal for those who want to use all the graphics capabilities built into the TRS-80. A tutorial method is used with many demonstrations. It is based on the Level I, but all material is suitable for Level II use. $8.95*

● **TOOLS & TECHNIQUES FOR ELECTRONICS**—BK7348—by A. A. Wicks is an easy-to-understand book written for the beginning kit builder as well as the experienced hobbyist. It has numerous pictures and descriptions of the safe and correct ways to use basic and specialized tools for electronic projects as well as specialized metal working tools and the chemical aids which are used in repair shops. $4.95*

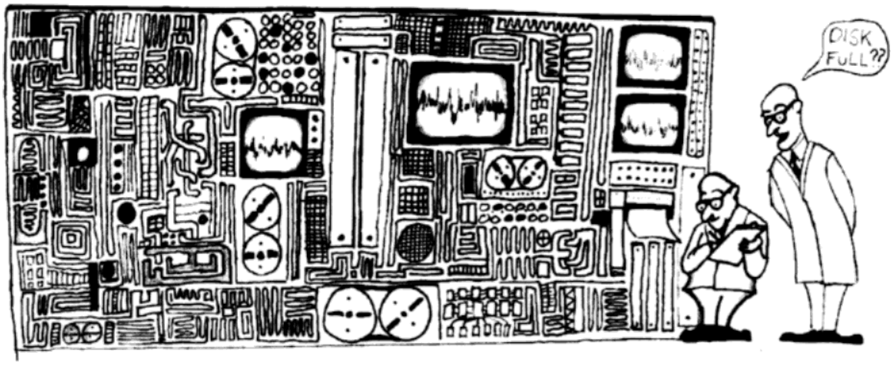
● **HOW TO BUILD A MICROCOMPUTER—AND REALLY UNDERSTAND IT**—BK7325—by Sam Creason. The electronics hobbyist who wants to build his own microcomputer system now has a practical "How-To" guidebook. This book is a combination technical manual and programming guide that takes the hobbyist step-by-step through the design, construction, testing and debugging of a complete microcomputer system. Must reading for anyone desiring a true understanding of small computer systems. $9.95*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to 80 Microcomputing Bookshelf ● Peterborough NH 03458. Be sure to include check or detailed credit card information.

No C.O.D. orders accepted. All above add $1.50 handling. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

PRICES SUBJECT TO CHANGE WITHOUT NOTICE

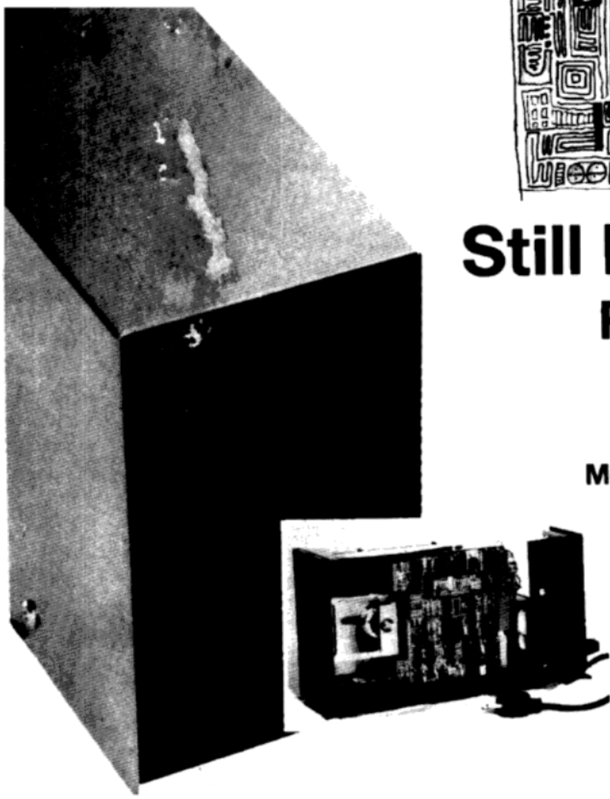


Still have storage problems?

Fully TRS-80® Compatible

(Model I or III)

Disk Drives



Model B/51 250K, 1 Side, 40 Tracks............ **$321.00**
B/52 500K, 2 Sides, 40 Tracks per side...... $439.00
B/91 500K, 1 side, 80 Tracks............... $439.00
B/92 1 Meg, 2 Sides, 80 Tracks per side..... $590.00

Includes: Case and Power Supply — Fully Tested
10 Day Money Back Guarantee
90 Days Parts and Labor Warranty (Excl. Shpg.)

Here's just a few of the features of our Disk Drives

- 40/80 Tracks
- Auto-eject
- Hi-Temp stability
- Speed constant ‹1½%
- Single/double density
- Optical sensors—no switches

ADDS MORE POWER TO YOUR SYSTEM

Save time . . . Order by phone

Orders only: 1-800-621-3229
Information: 312-987-1024
Tech. assistance: 312-987-1032

Other Money Savings Opportunities Order by Phone or Mail

CABLES
2 Drive Cable....... **$29.00**
4 Drive Cable....... **$39.00**

Diskettes
Verbatim.... 10 for **$31.00**
Syncom...... 10 for **$35.00**
Plastic File Box...... **$3.95**

Operating Systems
TRSDOS 2.3....... **$14.95**
40 Track Patch....... **$9.95**
NEWDOS + 40 Track......... **$99.00**
Newdos 80....... **$139.00**
TRSDOS Manual..... **$5.95**
Super Utility....... **$49.00**
Make 80........... **$14.00**

TRS-80
16K Model III..... **$899.00**
48K Model III w/2 Disks....... **$1900.00**
Expansion Interface OK.............. **$274.00**

Printers
Centronics 737-1.. **$815.00**
IDS 460......... **$1219.00**
Epson MX80...... **$550.00**
Okidata M80...... **$499.00**
Microline 82..... **$650.00**
Microline 83..... **$950.00**
Okidata prices include tractors
NEC Spinwriter 5530 (freight collect).. **$2579.00**

16K Memory Kits
Prime NEC 200ns dynamic RAM. Comes with complete instructions........ **$39.00**
MPI Service Manual............ **$3.00**
MPI Engineering Manual........... **$30.00**

MIDWEST COMPUTER PERIPHERALS® ✓8

1467 S. MICHIGAN
CHICAGO, ILLINOIS 60605

TRS 80 TANDY CORP

FREE! UPS GROUND SHIPPING ON ALL ORDERS OVER $100.00

Quantity	Description	$ each	Total
		6% IL Tax	
		TOTAL	

(Minimum order $50.00)

☐ Check enclosed
Bill my ☐ Visa ☐ Am Ex
☐ Master Charge
Acct. No. ________ Exp. ________

Name ________
Address ________
City ________ State ________ Zip ________

Not Responsible for Typographical Errors Prices Subject to Change w/o notice

Why use their flexible discs:

Ampex, Athana, BASF, Caelus, Control Data, Dysan, IBM, Inmac, K-Line, Maxell, Nashua, Scotch, Shugart, Syncom, 3M, Verbatim or Wabash

when you could be using

MEMOREX

for as low as $1.99 each?

Find the flexible disc you're now using on our cross reference list... then write down the equivalent Memorex part number you should be ordering.

Product Family	Product Description	Memorex Part Number (3201-)	CE quant. 100 price per disc ($)	Ampex	Athana	BASF	Caelus	Dysan	IBM	K-Line	Maxell	Nashua	Scotch 3M	Shugart	Syncom	Verbatim	Wabash	Inmac	Control Data
	IBM Compatible (128 B/S, 26 sectors)	3060	2.19	5FD-111110	473071	53428	CM-F11	800506	2305830	40013	FD1-128	FD-1	740-0	S/A 100	15002	FD34-1000	F111111X	7870-K	421602
Flexible Disc 1s	IBM Compatible (128 B/S, 26 sectors) w/ W.P.N.	3062	2.24	–	–	–	–	–	–	–	–	–	740-0	–	–	–	–	–	–
	IBM Compatible (128 B/S, 26 sectors) w/ W.P.N. & Hub ring	3064	2.55	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Single Headed Drives	IBM Compatible (128 B/S, 26 sectors) REVERSIBLE	1729	3.35	5FD-113110	473072	54431	–	–	–	40015	–	FD-2	740/2-0	–	15150	FF34-2000	F171111X	7860-K	–
Single Density Media	IBM System 6 Compatible	3066	2.19	–	473077	54561	–	800509	1669959	40014	–	–	740-0 0S6	–	15003	FD60-1000	F116111X	–	–
	IBM Compatible (256 B/S, 15 sectors)	3109	2.19	5FD-111210	473073	–	–	800584	2305845	40040	–	–	740-3600	–	15005	FD36-1000	F112111X	7861-K	–
	IBM Compatible (512 B/S, 8 sectors)	3110	2.19	–	473074	–	–	800585	1669954	40044	–	–	–	–	15004	FD60-1000	F113111X	7889-K	–
	Shugart Compatible, 32 hard sector	3015	2.19	5FD-211010	470901	53802	CM-F21	101/1	–	40016	FH1-32	FD-132	740-32	S/A 101	15025	FD32-1000	–	7890-K	421322
	Shugart Compatible, 32 hard sector REVERSIBLE	3025	3.35	5FD-213010	–	–	–	–	–	40017	–	–	740/2-32	–	15151	FF32-2000	–	7880-K	–
	Wang Compatible, 32 hard sector w/Hub ring	3087	2.50	–	–	54491	–	–	–	–	–	–	740-32RM	–	–	–	F37A411X	–	–
	CPT 8000 Compatible	3045	2.79	–	–	–	–	–	–	–	–	–	–	–	15226	–	–	–	–
Flexible Disc 1d	IBM Compatible (128 B/S, 26 sectors)	3090	2.95	5FD-121010	474071	54566	–	3740/1D	–	40047	FD1-128/M2100	FD-1D	741-0	–	–	FD34-8000	F131111X	7857-K	423002
	Soft Sector (128 B/S, 26 sectors) REVERSIBLE	3093	3.99	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Single Headed Drives	Shugart Compatible, 32 hard sector	3091	2.95	5FD-221010	470801	54596	–	101/1D	–	40024	FH1-32D	–	741-32	S/A 103	15075	FD32-8000	F33A410X	7887-K	423322
Double Density Media	Shugart Compatible, 32 hard sector REVERSIBLE	3094	3.99	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Wang Compatible, 32 hard sector w/Hub ring	3088	3.20	–	–	–	–	–	–	–	–	–	–	–	–	–	F22A411X	–	–
Flexible Disc 2s	Soft Sector (Unformatted)	3101	3.84	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Soft Sector (128 B/S, 26 sectors)	3113	3.84	–	–	54428	–	800814	1766870	–	–	–	–	S/A 150	15153	FD10-4026	F121111X	–	–
Double Headed Drives	Soft Sector (256 B/S, 15 sectors)	3106	3.84	–	473477	54226	–	800815	2736700	40043	FD2-256D	–	742-0	–	15154	FD10-4015	F122111X	7856-K	424612
Single Density Media	32 Hard Sector	3108	3.84	–	–	–	–	–	–	–	FH2-32	–	–	–	–	–	–	–	–
Flexible Disc 2d	Soft Sector (Unformatted)	3102	3.49	–	473485	–	–	DY150	–	40028	FD2-XDM	FD-2D	743-0	–	15103	DD34-4001	–	–	425002
	Soft Sector (128 B/S, 26 sectors)	3115	3.49	–	–	–	–	–	–	–	–	–	–	S/A 150	–	–	–	–	–
Double Headed Drives	Soft Sector (256 B/S, 26 sectors)	3103	3.49	–	473471	54325	–	800817	1766872	40019	FD2-2560	–	743-0/256	–	15101	DD34-4026	F144111X	78[illegible]3-K	425[illegible]
Double Density Media	Soft Sector (512 B/S, 15 sectors)	3114	3.49	–	473472	54479	–	800818	1669044	40039	–	–	743-0/512	–	15100	DD34-4015	F145111X	–	42[illegible]
	Soft Sector (1024 B/S, 8 sectors)	3104	3.49	–	473473	54485	–	800819	1669045	40020	–	–	743-0/1024	–	15102	DD34-4008	F147111X	7853-K	[illegible]
	32 Hard Sector	3105	3.49	5FD-321010	470851	–	–	101/2D	–	40021	FH2-32D	–	743-32	S/A 151	15125	DD32-4000	F34A411X	7881-K	[illegible]
	Burroughs B-80 Compatible 32 Hard Sector	3092	3.49	–	–	–	–	–	–	–	–	–	–	–	–	–	F34A611X	–	[illegible]
	Soft Sector (1024 B/S, 8 sectors) w/ Hub Ring	3116	3.75	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	[illegible]
Flexible Disc FD Memorex 651 or Equiv Drive Compatible	FD VI (Vinyl Jacket)	30712003	2.95	–	470651	–	CM-F31	FDIV	–	40002	–	FD-165	511-0	–	15026	FD65-1000	F61A111X	7910	–
Mini Flexible Disc 1s	Soft Sector (Unformatted)	3401	1.99	–	475001	54256	–	104/1	–	40500	MD1	MD-1	744-0	S/A 104	15300	MD525-01	M11A211X	7897	441002
5¼ Single-Headed Drives	10 Hard Sector	3403	1.99	–	475010	54257	–	107/1	–	40501	–	MD-110	744-10	S/A 107	15325	MD525-10	M41A211X	7898	441102
Single-Density Media	16 Hard Sector	3405	1.99	–	475016	54258	–	105/1	–	40502	MH1	MD-116	744-16	S/A 105	15326	MD525-16	M51A211X	7899	441162
	Soft Sector (Unformatted) W/Hub Ring	3431	2.19	–	–	–	–	–	–	–	–	–	–	–	–	MD525-01	–	–	–
	10 Hard Sector w/Hub Ring	3433	2.19	–	–	–	–	–	–	–	–	–	–	–	–	MD525-10	–	–	–
	16 Hard Sector w/Hub Ring	3435	2.19	–	–	–	–	–	–	–	–	–	–	–	–	MD525-16	–	–	–
Mini Flexible Disc 1d	Soft Sector	3417	2.24	–	–	54646	–	104/1D	–	–	–	–	–	–	–	MD540-01	–	–	–
5¼ Single Headed Drives	10 Hard Sector	3418	2.24	–	–	54649	–	107/1D	–	–	–	–	–	–	–	MD540-10	–	–	–
Double Density Media	16 Hard Sector	3419	2.24	–	–	54652	–	105/1D	–	–	–	–	–	–	–	MD540-16	–	–	–
Mini Flexible Disc 2d	Soft Sector	3421	2.74	–	–	54624	–	104/2D	–	–	MD2-D	–	745-0	S/A 154	–	MD550-01	–	–	–
5¼ Double-Headed Drives	10 Hard Sector	3423	2.74	–	–	54627	–	107/2D	–	–	–	–	745-10	S/A 157	–	MD550-10	–	–	–
Double Density Media	16 Hard Sector	3425	2.74	–	–	54630	–	105/2D	–	–	MH2-D	–	745-16	S/A 155	–	MD550-16	–	–	–

Memorex Flexible Discs...The Ultimate in Memory Excellence

Quality
Memorex means quality products that you can depend on. Quality control at Memorex means starting with the best materials available. Continual surveillance throughout the entire manufacturing process. The benefit of Memorex's years of experience in magnetic media production, resulting, for instance, in proprietary coating formulations. The most sophisticated testing procedures you'll find anywhere in the business.

100 Percent Error Free
Each and every Memorex Flexible Disc is certified to be 100 percent error free. *Each* track of *each* flexible disc is tested, individually, to Memorex's stringent standards of excellence. They test signal amplitude, resolution, low-pass modulation, overwrite, missing pulse error and extra pulse error. They are torque-tested, and competitively tested on drives available from almost every major drive manufacturer in the industry including drives that Memorex manufactures. Rigid quality audits are built into every step of the manufacturing process and stringent testing result in a standard of excellence that assures you, our customer, of a quality product designed for increased data reliability and consistent top performance.

Customer-Oriented Packaging
Memorex's commitment to excellence does not stop with a quality product. They are proud of their flexible discs and they package them with pride. Both their packaging and their labeling have been designed with your ease of identification and use in mind. The desk-top box containing ten discs is convenient for filing and storage. Both box labels and jacket labels provide full information on compatibility, density, sectoring, and record length. Envelopes with multi-language care and handling instructions and color-coded removable labels are included. A write-protect feature is available to provide data security.

Full One Year Warranty — Your Assurance of Quality
Memorex Flexible Discs will be replaced free of charge by Memorex if they are found to be defective in materials or workmanship within one year of the date of purchase. Other than replacement, Memorex will not be responsible for any damages or losses (including consequential damages) caused by the use of Memorex Flexible Discs.

Quantity Discounts Available
Memorex Flexible Discs are packed 10 discs to a carton and 10 cartons to a case. Please order only in increments of 100 units for quantity 100 pricing. We are also willing to accommodate your smaller orders. Quantities less than 100 units are available in increments of 10 units at a 10% surcharge. **Quantity discounts** are also available. Order 500 or more discs at the same time and deduct 1%; 1,000 or more saves you 2%; 2,000 or more saves you 3%; 5,000 or more saves you 5%; 10,000 or more saves you 7%; 25,000 or more saves you 8%; 50,000 or more saves you 9% and 100,000 or more discs earns you a 10% discount off our super low quantity 100 price. Almost all Memorex Flexible Discs are immediately available from **CE**. Our warehouse facilities are equipped to help us get you the quality product you need, when you need it. If you need further assistance to find the flexible disc that's right for you, call the Memorex compatibility hotline. Dial 800-538-8080 and ask for the *flexible disc hotline* extension 0997. In California dial 800-672-3525 extension 0997.

Buy with Confidence
To get the fastest delivery from **CE** of your Memorex Flexible Discs, send or phone your order directly to our Computer Products Division. Be sure to calculate your price using the **CE** prices in this ad. Michigan residents please add 4% sales tax. Written purchase orders are accepted from approved government agencies and most well rated firms at a 10% surcharge for net 10 billing. All sales are subject to availability. All sales are final. Prices, terms and specifications are subject to change without notice. Out of stock items will be placed on backorder automatically unless **CE** is instructed differently. International orders are invited with a $20.00 surcharge for special handling in addition to shipping charges. All shipments are F.O.B. Ann Arbor, Michigan. No COD's please. Non-certified and foreign checks require bank clearance.

Mail orders to: **Communications Electronics**, Box 1002, Ann Arbor, Michigan 48106 U.S.A. Add $8.00 per case or partial-case of 100 8-inch discs or $6.00 per case of 100 5¼-inch mini-discs for U.P.S. *ground* shipping and handling in the continental U.S.A. If you have a Master Charge or Visa card, you may call anytime and place a credit card order. **Order toll-free** in the United States. Call anytime 800-521-4414. If you are outside the U.S. or in Michigan, dial 313-994-4444. **Dealer inquiries invited.** All order lines at **Communications Electronics** are staffed 24 hours.

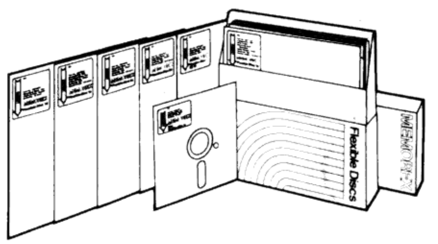
MEMBER DIRECT MAIL MARKETING ASSOCIATION

CODE OF ETHICS

VISA

master charge

Order Toll-Free!
(800) 521-4414



For Data Reliability—Memorex Flexible Discs

COMMUNICATIONS ELECTRONICS™ ✓263

Computer Products Division

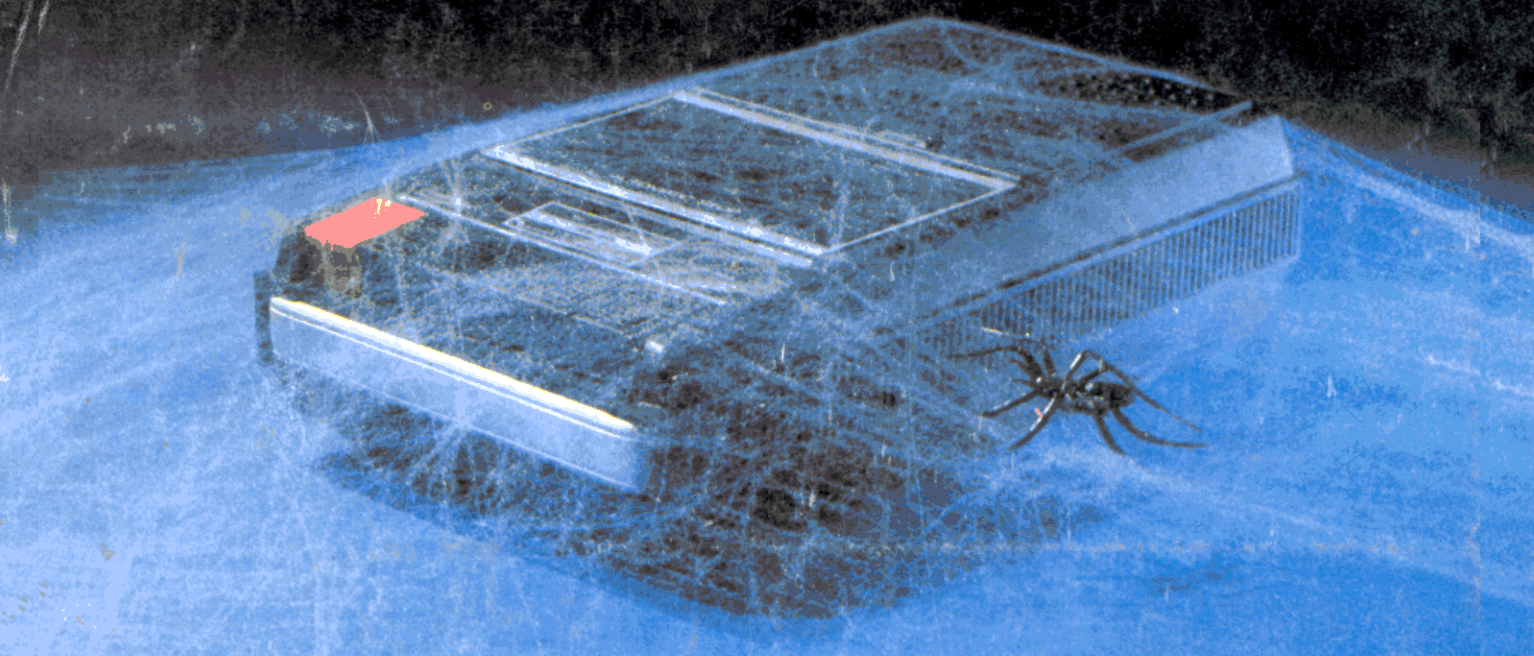
854 Phoenix ☐ Box 1002 ☐ Ann Arbor, Michigan 48106 U.S.A.
Call TOLL-FREE (800) 521-4414 or outside U.S.A. (313) 994-4444

LIST OF ADVERTISERS

For further information from our advertisers, please use the Reader Service card.

*This advertiser prefers to be contacted directly

TIRED OF WAITING?



Frustrating isn't it! No matter how much you speed up your program it still seems to take forever to save data onto a cassette. Wouldn't it be great if someone could design a mass storage system with the speed of a disk, but at half the cost? *Exatron* did, the *Exatron Stringy Floppy (ESF)*.

Totally self-contained, the ESF is an extremely fast, reliable, and economical alternative to cassette or disk storage of programs or data. All of the ESF's operations are under the computer's control, with no buttons, switches, knobs or levers to adjust or forget.

The ESF uses a miniature tape cartridge, about the size of a business card, called a wafer. The transport mechanism uses a direct drive motor with only one moving part. Designed to read and write digital data only, the ESF suffers from none of the drawbacks of cassettes - without the expense of disks.

Several versions of the ESF are available, for the *TRS-80, Apple, PET, OSI* and an *RS 232* unit. Even the slowest of the units is 15 times faster than a cassette, and all are as reliable as disk drives - in fact a lot of users say they are *more* reliable!

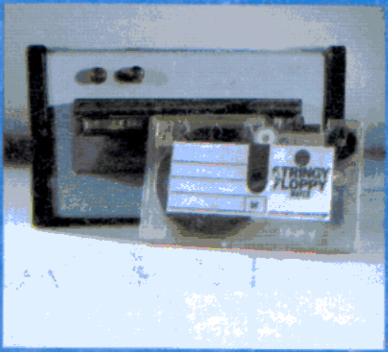
excellence in electronics

exatron

To get further information about the ESF give Exatron a call on their Hot Line *800-538 8559* (inside California 408-737 7111).

If you can't wait any longer then take advantage of their 30 day money-back guarantee, you've nothing to lose but time!

181 Commercial Street
Sunnyvale, CA 94086



3

TRS-80, Apple and PET are trademarks of Tandy, Apple and Commodore respectively.